“Diffusion Geometry Based Nonlinear Methods for Hyperspectral Change Detection”

Contract Item 0001AC – Final Technical Report
Plain Sight Systems, Inc.

Air Force Office of Scientific Research
Contract # FA9550-09-C-0189

Dr. Andreas Coppi (POC)
Dr. Frederick Warner
Plain Sight Systems, Inc.
19 Whitney Avenue
New Haven, CT 06510
(203) 285-8617
coppi@plainsight.com

Dr. Ronald. R. Coifman (POC)
Dr. Matthew Hirn
Program of Applied Mathematics
Yale University
PO BOX 208283
New Haven, CT 06520-8283
(203) 432-1213
coifman@math.yale.edu

May 12, 2010

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 12-05-2010	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* 20090715 - 20100414

4. TITLE AND SUBTITLE

"Diffusion Geometry Based Nonlinear Methods for Hyperspectral Change Detection"

5a. CONTRACT NUMBER

FA9550-09-C-0189

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Dr. Ronald Coifman
Dr. Andreas Coppi
Dr. Matthew Hirn
Dr. Frederick Warner

5d. PROJECT NUMBER

5e. TASK NUMBER

0001AC

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Plain Sight Systems, Inc. - 19 Whitney Avenue, New Haven, CT 06510
STTR University Partner: Program of Applied Mathematics, Yale University, P.O. Box 208283, New Haven, CT 06520-8283

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AF OFFICE OF SCIENTIFIC RESEARCH
875 NORTH RANDOLPH STREET ROOM 3112
ARLINGTON VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Throughout this Phase I project, we have integrated a suite of nonlinear signal processing algorithms derived from diffusion geometry into an existing proprietary Hyperspectral processing toolbox. These methods enable the organization and comparison of spatiospectral features of hyperspectral images acquired under different conditions, for target detection, change and anomaly assessment. The main ingredients in our approach involve a high level "geometrization" of spatio spectral signatures.
We developed an approach to simultaneously segment a scene in terms of similarities of spatio spectral signatures at different inference as well as a partition of the feature space of spectra and morphology into groups of features related to the various locations on the scene. We refer to this approach in which we interrogate and organize both the pixels and their responses as the questionnaire organization paradigm.. This spectral segmentation methodology is critical for change detection as it enables to isolate changes by comparing their relation to their spatio-spectral folders. The folder identity provides invariant features for change detection.

15. SUBJECT TERMS

Hyperspectral, Change Detection, Remote Sensing, Target Detection, Non-Linear, Processing

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Andreas Coppi
UU	UU	UU	SAR	137	**19b. TELEPHONE NUMBER** *(Include area code)* 203-285-8617

Summary of Results

Throughout this Phase I project, we have integrated a suite of nonlinear signal processing algorithms derived from diffusion geometry into an existing proprietary Hyperspectral processing toolbox. These methods enable the organization and comparison of spatio-spectral features of hyperspectral images acquired under different conditions, for target detection, change and anomaly assessment. The main ingredients in our approach involve a high level "geometrization" of spatio spectral signatures.

In the first months of this project (see Appendix A), we developed an approach to simultaneously segment a scene in terms of similarities of spatio spectral signatures at different inference as well as a partition of the feature space of spectra and morphology into groups of features related to the various locations on the scene. We refer to this approach in which we interrogate and organize both the pixels and their responses as the *questionnaire* organization paradigm described in Appendix A. This spectral segmentation methodology is critical for change detection as it enables to isolate changes by comparing their relation to their spatio-spectral folders. The folder identity provides invariant features for change detection.

We have also developed a tool which enables the automated search in the image (viewed as a data base) which operates as follows: The user identifies a number of reference pixels in the scene (or in other scenes), and obtains an image in which only pixels, having some affinity with the selected reference pixels, are displayed. Moreover as the affinity among spectra is defined through the use of a local Mahalanobis distances, this tool can find related pixels across various acquisitions. We have tested this methodology for matching biological spectra across a data base of hyperspectral pathology slides acquires with different instruments in different conditions, as well as on hyperspectral images of Smith Island acquired on two different days for change detection.

The same methodology also extracts automatically independent components of the spectrum building an empirical model of the constituents of the scene. It is precisely through this model that most efficient target search and change detection can be performed.

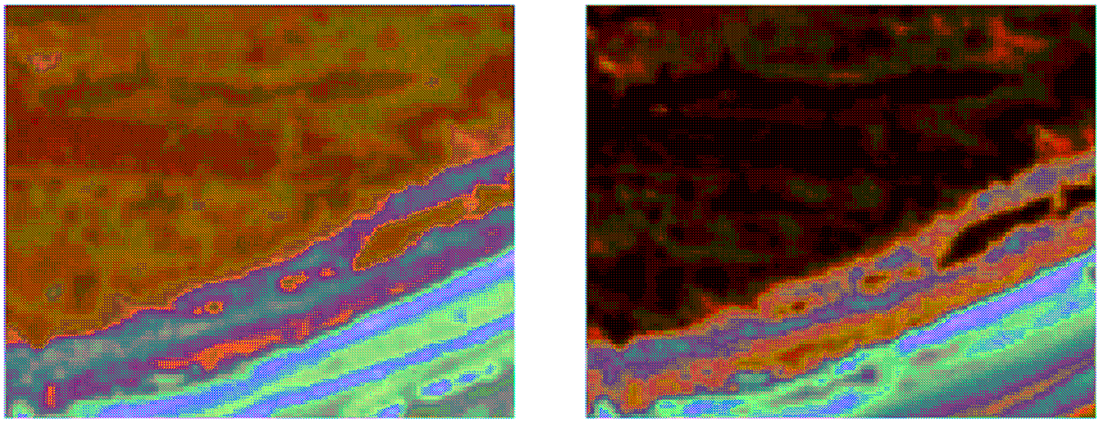
Plain Sight Systems, Inc. has collaborated with Dr. Coifman's group in the Applied Mathematics Department of Yale University for algorithm development, and has integrated these methods into their proprietary *Hyperspectral Explorer* software package for hyperspectral information organization and processing (Appendix D).

The main software development tasks have been finished and we are currently applying and validating the methods to hyperspectral data of interest to the Air Force.

The enclosed reports (Appendix B & C) describe some of the algorithms that were tested and developed in the latter months of Phase I. It is expected that they will be simplified considerably and enable real time change detection. We have mostly tested a variety of scaling and renormalizations that would stay invariant across instruments and illumination conditions.

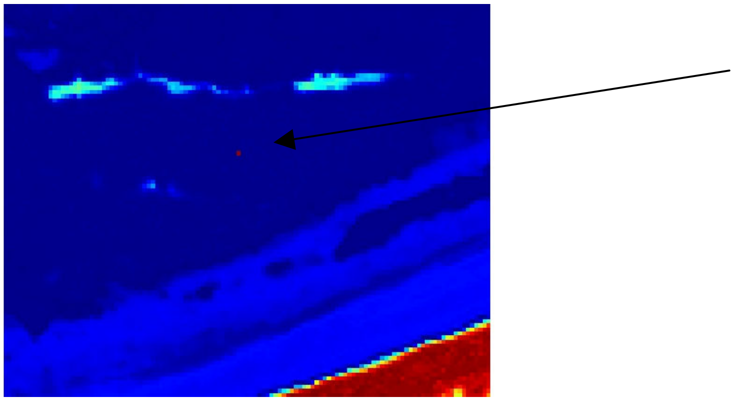
In the first report on anomaly and target detection (Appendix B), we have viewed the local spectral covariance matrix as a good simple spectral invariant because it quantifies the relation between a location to its most spectrally similar and spatially close points. As shown, this approach is quite effective in isolating anomalous pixels. In the second enclosed report (Appendix C), we introduce a number of mathematical calibration methods to relate calibrated spectra on two days for change detection. Of course, this method can easily be combined with the target detection approach of the first report.

As an illustration of results we look at an area near the beach on two different days (all spectra are different). In the image below two daily images, a pixel indicated as red below has been modified in its spectrum and is actually the only one detected, except for the red area off the beach to the lower right which is simply change due to ocean wave activity.



(a) Day 1

(b) Day 2



Summary of Theory

To summarize, an existing signal processing toolbox was augmented by methods developed in Phase I to extract structure and information from heterogeneous images and other data sets. These methodologies enable efficient integration and fusion of heterogeneous image sources with information processing tasks and are particularly well adapted to hyperspectral imagery in which spectral content can be integrated with geometric image features.

The approach uses the network of inferences and similarities between the data points to create robust nonlinear estimators for missing or noisy entries. This method enables coherent analysis of data from a multiplicity of sources generalizing signal processing to a nonlinear setting. By building empirical data models it achieves nonlinear decorrelation and dimensionality reduction for intrinsic data structures.

We start by discussing feature based filtering and signal processing on graphs as a simple way to understand the effect of introducing affinity (similarity) based diffusions on image data. For simplicity of display, we start by considering a regular gray level image in which we associate to each pixel p a vector $v(p)$ of features. For example, a multiband electromagnetic spectrum, a filter bank, or the simplest of all, a 5x5 subimage centered at the pixel, or any combination of features as above. Define a Markov filter

$$A_{p,q} = \frac{\exp(-\left\| v(p) - v(q) \right\|^2 / \varepsilon)}{\sum_q \exp(-\left\| v(p) - v(q) \right\|^2 / \varepsilon)}$$

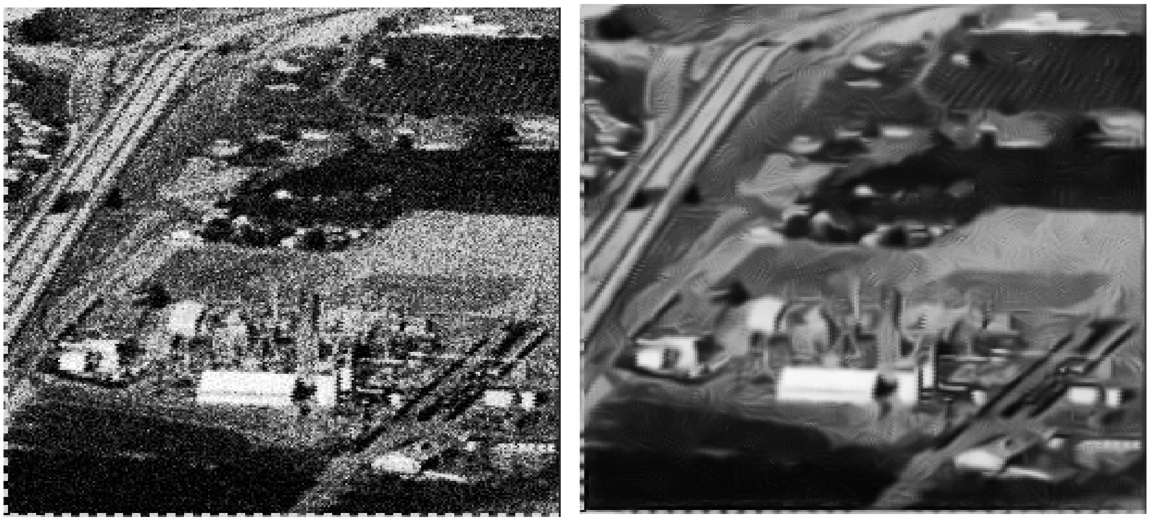
The image $I(q)$ below was filtered using the (nonlinear in the features) procedure described above where the feature vector is the 5x5 patch around a pixel.

$$\bar{I}(p) = \sum_q A_{p,q} I(q) = \sum_q \frac{\exp(-\left\| v(p) - v(q) \right\|^2 / \varepsilon)}{\sum_{q'} \exp(-\left\| v(p) - v(q') \right\|^2 / \varepsilon)} I(q)$$

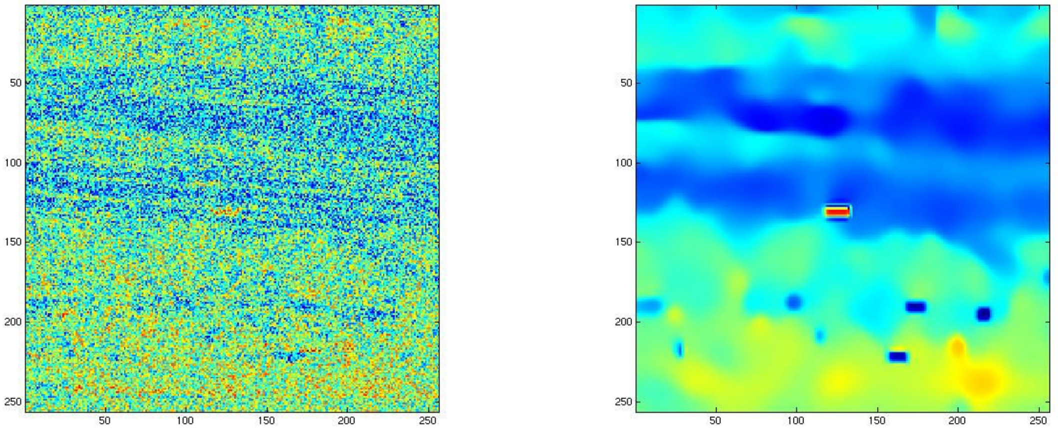
Observe that the edges are well preserved, as patches translated parallel to an edge are similar and contribute more to the averaging procedure. We should also observe that if we were to repeat the procedure on the filtered image we would get a numerical implementation of various nonlinear heat diffusions for image processing as done by Osher and Rudin.

(It is useful to replace A by a Bi-Markovian version $A_{p,q} = \dfrac{\exp(-\left\| v(p) - v(q) \right\|^2 / \varepsilon)}{\omega(p)\omega(q)}$,

where the weights are selected so that A is Markov in p and q.)



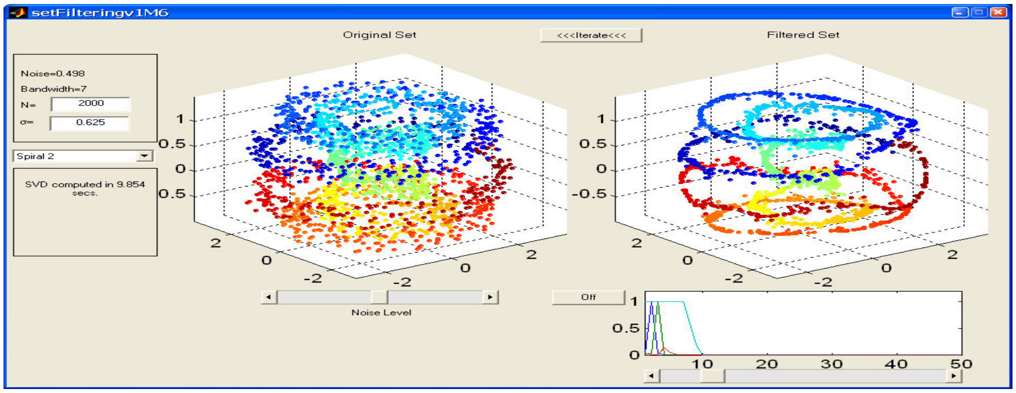
The noisy IR image below was filtered using a vector of 25 statistical features associated with each pixel



The Markov matrix used for filtering, defines a diffusion on the Graph of patches or features viewed as a subset of 25 dimensional Euclidean space. The eigenvectors of this diffusion permit us to compute all of its powers and to define a multiscale diffusion geometry and signal processing on this "image graph". (By viewing the image as a function on its feature graph, in which $v(p)$ are vertices and $A_{p,q}$ are the weights of the edge between $v(p)$ and $v(q)$, we can analyze the image relative to its features.)

For the next example, consider 3 noisy sensors measuring the xyz-coordinates of a trajectory in three dimensions .We could try to denoise each coordinate separately. Or use the position vector as a feature vector as we did for the images above (in our case each sensor could be the intensity of each spectral band and the denoising is achieved only through comparison with location in the scene with similar spectra).

The construction above should be viewed as signal processing, filtering, on the data graph. We view all points of the trajectory as a data graph, i.e. data points p and q are vertices and $A_{p,q}$ is the weight of the edge connecting them measuring their similarity or affinity at the smallest scale. We consider the eigenvectors of the Markov matrix $A_{p,q}$ defined above as a basis for all functions on this graph. We can then expand each coordinate as a function on that graph, and restrict the expansion to the first few "low frequency" eigenfunctions, i.e. filter it and use the filtered coordinates as a clean trajectory. This generalizes the simple filtering done on images above see figure 3 below.


setFilteringv1M6

Noise=0.498
Bandwidth=7
N= 2000
σ= 0.625

Spiral 2

SVD computed in 9.854 secs.

Fig 3. The green, red and blue curves are respectively the coefficients of the xyz coordinates, as filtered above, using less than 10 eigenvectors of the Markov matrix.

Diffusion Geometries

These simple examples indicate that diffusion and harmonic analysis are useful for coherent sensor integration and fusion, enabling signal processing for nonlinearly correlated data streams. Diffusion geometries enable the definition of affinities and related scales between any digital data points in R^n (provided of course that the "infinitesimal proximity" in the coordinates corresponds to true affinity between data points). Moreover it enables the organization of the population of sensor output into "affinity folders" or subsets, at different scales, with a high level of affinity among their responses.

In particular the eigenfunctions of the Diffusion operator or equivalently a Laplacian on a graph provide useful empirical coordinates, which enable an embedding of the data to low dimensional spaces so that the diffusion distance at time t on the original data becomes Euclidean distance in the embedding, in effect providing a nonlinear version of the SVD as well as a powerful dimensional reduction methodology. Moreover we indicate how the diffusion at different times leads to a multiscale analysis generalizing wavelets and similar scaling mechanisms.

To be specific, let the bi-Markov matrix A defined above be represented in terms of its eigenvectors, and define the diffusion map at time t into m dimensional Euclidean space by

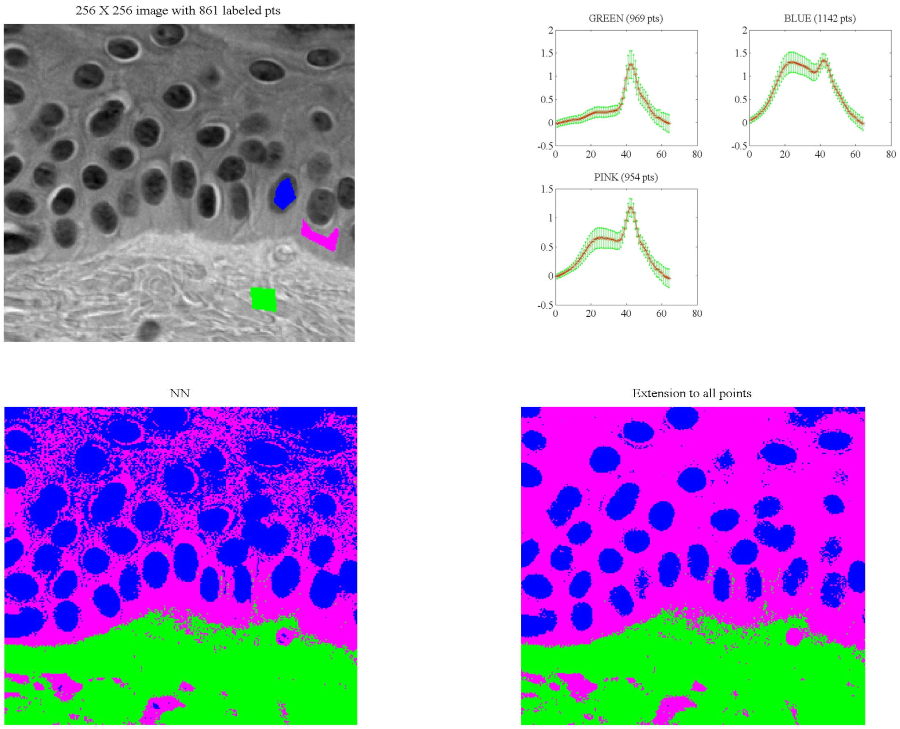
$$A_{p,q} = \sum \lambda_l^2 \phi_l(X_p)\phi_l(X_q)$$
$$X_p \rightarrow (\lambda_1^{\,t}\phi_1(X_p), \lambda_2^{\,t}\phi_2(X_p), \lambda_3^{\,t}\phi_3(X_p),.., \lambda_m^{\,t}\phi_m(X_p)) = \widetilde{X}_p^{\,t}$$

For a given t, we determine m so that $\lambda_{m+1}^{\,t}$ is negligible. The diffusion distance at time t between X_p and X_q is given as

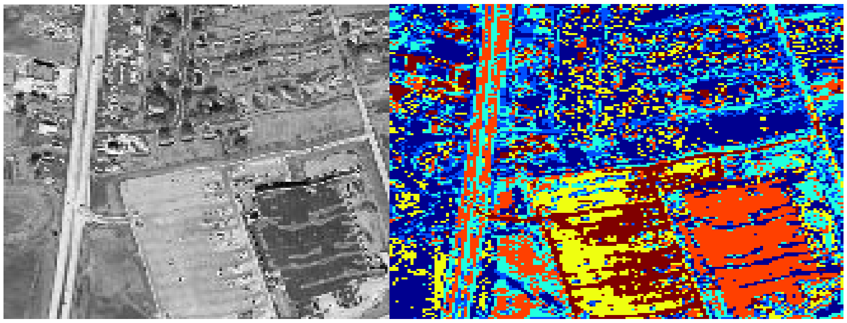
$$d_t^{\,2}(p,q) = A_{p,p} + A_{q,q} - 2A_{p,q} = \sum \lambda_l^{2t}(\phi_l(X_p) - \phi_l(X_q))^2 = \left\| \widetilde{X}_p^{\,t} - \widetilde{X}_q^{\,t} \ \right\|^2$$

As an application we show the advantage of using diffusion distance over ordinary Euclidean distance, in the context of hyperspectral imagery and tracking. The hyperspectral image below has three sets, labeled as green, red blue. The task is to classify the rest of the image by spectral similarity to the labeled sets. A nearest neighbor classifier using Euclidean distance fails due to regional drift in spectra while the diffusion distance measuring all chains of similarity linking a given pixel to the labeled set does remarkably well.

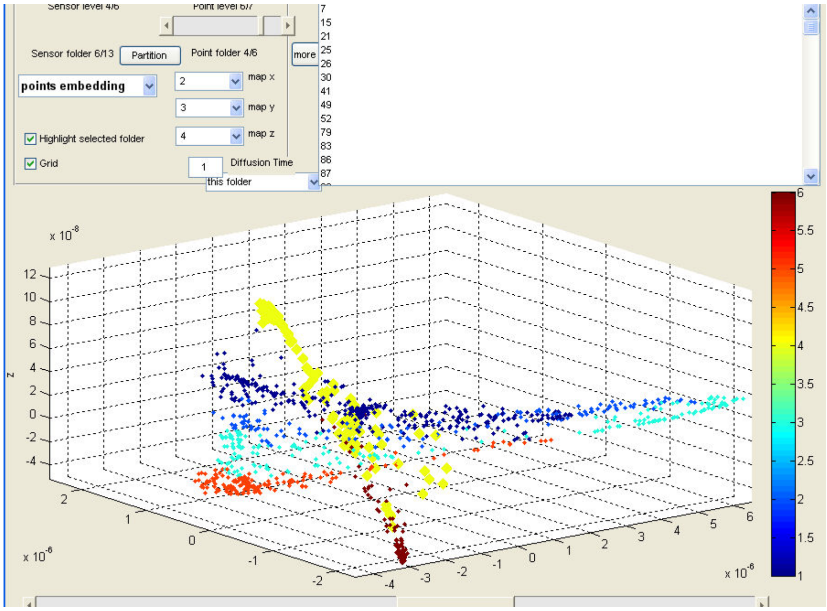
Conventional nearest neighbor search , compared with a diffusion search. The data is a pathology slide ,each pixel is a digital document (spectrum below for each class)



The diffusion map enables us to represent geometrically an abstract set of measurements on a sensor array (measurement space) and measure directly diffusion distances in the low dimensional representation. Diffusion geometry enables a multiscale organization (in feature space) of pixels in a scene as illustrated in the following urban scene



Where the left is the original image while the right is spectrally segmented. This method identifies not sensitive to the selection of bands or to illumination and atmospheric distortions, thereby providing invariant features. The segmentation above is obtained by clustering at different scales in the diffusion embedding space as seen below



Sensor level 4/6

Point level 6/7

Sensor folder 6/13

Point folder 4/6

points embedding

2 map x

3 map y

4 map z

☑ Highlight selected folder

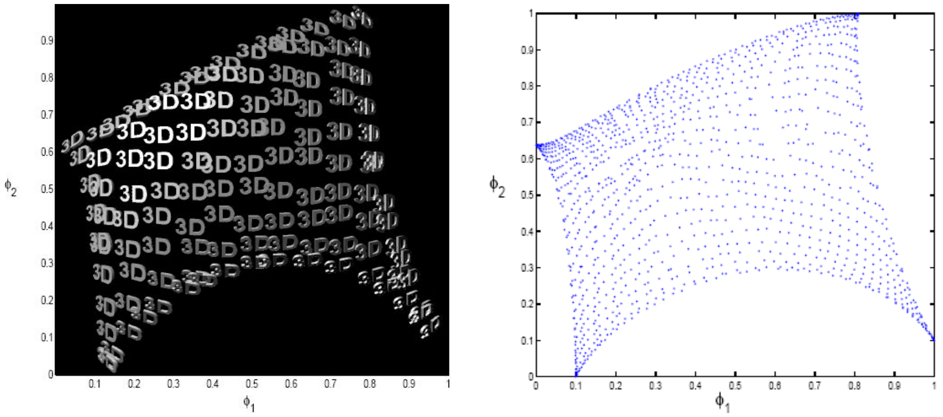
☑ Grid

1 Diffusion Time

this folder

Another methodology for defining invariant spectral features will be described as follows.

The next images illustrates the organizational ability of the diffusion maps on a collection of images of the text “**3D**” imaged in various random orientations relative to the camera and light as reordered by the diffusion mapping given by the first two nontrivial eigenfunctions. We see that the intrinsic geometry has emerged automatically.



.

Intrinsic parameterizations of hyperspectral images

As discussed previously it is possible to mathematically reparameterize spectral data to enable robust structural change detection. This procedure extends the methodologies described in [9]-[12] where PCA is used to renormalize spectral data to enable comparison between two acquisitions. Here we use local whitening to build a global explicit parameterization invariant under nonlinear perturbations of the spectrum, calibrating the data independently of the measurement modes (or even sensors, provided that they are affected by the same parameters).

More specifically, we let

$$A_{p,q} = \frac{\exp(\left\langle C_p^{-1}(\sigma_p - \sigma_p)\cdot(\sigma_p - \sigma_p)\right\rangle + \left\langle C_q^{-1}(\sigma_p - \sigma_p)\cdot(\sigma_p - \sigma_p)\right\rangle}{\omega(p)\omega(q)}$$

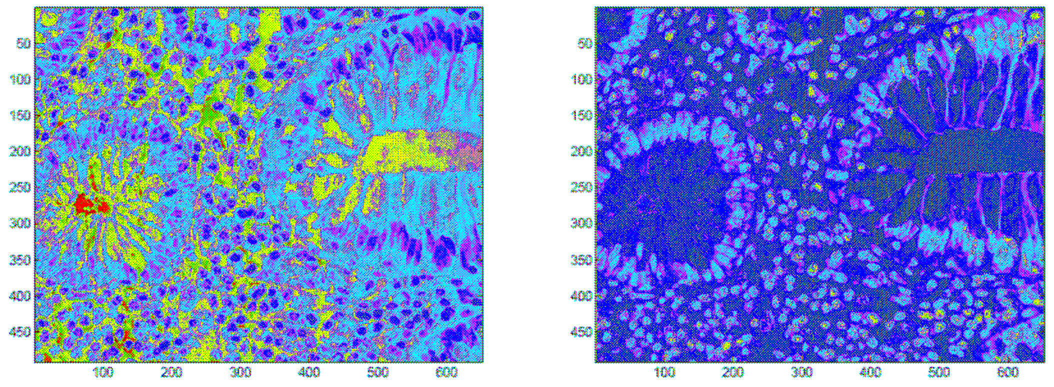
Where C_p^{-1} is the inverse of the covariance matrix of the spectra σ in a small region around p, and σ_p is a spectrum at p (or another feature vector which , such as, all 9 spectra in 3x3 block centered at p)

If we assume a generalized nonlinear Beer’s law i.e. that

$$\sigma_p = F(c_1, c_2, ..c_r) = F(c)$$

where the vector c represents the concentration (or “amount”) vector of *r* independent constituents affecting the measured spectrum. Then we can find among the first few eigenvectors of the matrix *A*, *r* monotone functions of the *r* independent constituents in *c*, (the underlying mathematical assumption is that the inverse of the covariance matrix is the “square “ of the Jacobian of the inverse of *F* from data to parameter *c*). We show that the process of computing the eigenvectors of *A* solves a nonlinear differential equation going from data to intrinsic parameters. Moreover we provide methodologies for extending these eigenvectors $\varphi_l(\sigma_p)$ from the measured data to any vector σ in the same dimension.

This construction is a natural nonlinear generalization of principal component analysis as well as independent component analysis. The remarkable property of this construction is that any random encoding of σ, say by projecting the spectra on a random collection of *r +1* vectors leads to the same parameterization of the independent constituents (with high probability), thereby reducing the complexity of acquisition to the number of relevant parameters, as opposed to full resolution spectra. In the picture below we show that classification of tissue type can easily be achieved through a small number of encoded light measurements



The image on the left is an RGB representation of the encoded measurements; whereas on the right, they have been organized to by diffusion geometry to provide intrinsic biological parameters quantifying tissue constituents. The image on the right is independent of the selection of spectral encodings.

Fusion of spatial and spectral information

For change detection in a heterogeneous environment, we can view each set of features as corresponding to different sensor, say spatial features, or spectral features, each category of features can be parameterized and normalized in its intrinsic diffusion coordinates. A new graph is then created combining the relevant diffusion coordinates emanating from different species of features as coordinates. This goal can also be achieved through direct concatenation of spectral patches in the scene, the point being that the spatial distribution of spectra may be a more robust indicator of change than single pixel-by-pixel match. Moreover this methodology also provides a powerful registration tool as it enables the matching of distributed spatio spectral features in both images independently of their position.

References

1. Belkin, M., & Niyogi, P. (2001). “Laplacian eigenmaps and spectral techniques for embedding and clustering.” Advances in Neural Information Processing Systems 14 (NIPS 2001) (p. 585).

2. Belkin, M., & Niyogi, P. (2003a). „Laplacian eigenmaps for dimensionality reduction and data representation.” Neural Computation, 6, 1373-1396.

3. Coifman, R. R., Lafon, S., Lee, A., Maggioni, M.,Nadler, B., Warner, F., & Zucker, S. (2005a). “Geometric diffusions as a tool for harmonic analysis and structure defnition of data. Part I: Diffusion maps.” Proc. of Nat. Acad. Sci., 7426-7431.

4. Coifman, R. R., Lafon, S., Lee, A., Maggioni, M.,Nadler, B., Warner, F., & Zucker, S. (2005b). “Geometric diffusions as a tool for harmonic analysis and structure defnition of data. Part II: Multiscale methods.” Proc. of Nat. Acad. Sci., 7432-7438.

5. Coifman RR, Lafon S: “Diffusion maps.” Appl Comp Harm Anal 2005, accepted for publication. “Introduction to diffusion maps and its applications.”

6. Coifman RR, Maggioni M: “Diffusion wavelets.” Tech. Rep. YALE/DCS/TR-1303, Appl Comp Harm Anal 2004, accepted for publication. “In-depth presentation of the multiscale construction of diffusion geometries, for multiscale analysis on graphs and manifolds.”

7. Coifman RR, Lafon S, Lee AB, Maggioni M, Nadler B, Warner FJ, Zucker SW: “Geometric diffusions as a tool for harmonic analysis and structure definition of data. Part I: Diffusion maps.” Proc Nat Acad Sci 2005, 102: 7426-7431. “Introduction to geometric diffusions with applciations to analysis of data sets and simulated physical systems.”

8. Coifman RR, Singer A. “Nonlinear Independent Component Analysis with Diffusion Maps.” Appl. Comput. Harmon. Anal. 25 (2008) 226–239.

9. A. Schaum and A. Stocker, “Hyperspectral change detection and supervised matched filtering based on covariance equalization,” Proceedings of the SPIE, vol. 5425, pp. 77-90 (2004).

10. A. Schaum and A. Stocker, “Linear chromodynamics models for hyperspectral target detection,” Proceedings of the IEEE Aerospace Conference (February 2003).

11. A. Schaum and A. Stocker, “Linear chromodynamics models for hyperspectral target detection,” Proceedings of the IEEE Aerospace Conference (February 2003).

12. H. Kwon and N. Nasrabadi, “Kernel RX Algorithm, A Nonlinear Anomaly Detector for Hyperspectral Imagery,” IEEE Transactions on Geoscience and Remote Sensing (February 2005).

Appendix A: Hyperspectral Image Organization Through Spectral "Questionnaires" (PowerPoint Presentation)

AFOSR STTR Report
October 2009

Hyperspectral image organization through spectral "questionnaires"

Plain Sight Systems

- The questionnaire approach is being tested to extract anomalies and change between two hyperspectral acquisitions of the same scene. After local Mahalanobis whitening, we build a joint questionnaire model for both images, and look for pixels whose response are inconsistent.

- This approach facilitates registration as the pixels’ features are organized independently of their location.

- We have developed an extension of our *Hyperspectral Explorer* package which implements the questionnaire approach on hyperspectral images, providing automatic segmentation and anomaly extraction capabilities.

- A hyperspectral search projection has been developed in which, after the selection specific reference spectra, pixels with related spectra in a scene are displayed by degree of relevance.

We use diffusion geometries as a tool to organize hyperspectral data or more generally vector valued images, or any digital database.

While Euclidean distance is useful in low dimensions to measure similarity between points, it fails in higher dimension.

Diffusion , or inference geometries, use Euclidean distance only when points are very close to each other to create affinity links between data points. These links build a Markov graph for which the distance between any two points is measured in terms of a probability of transition between them.

The eigenvectors of the Markov matrix, integrate all the local information and permit an embedding of the graph in lower dimension (diffusion maps).

We can use this geometry to organize our data– a collection of points in high dimensions, i.e. spectra of individual pixels– into a tree of folders.

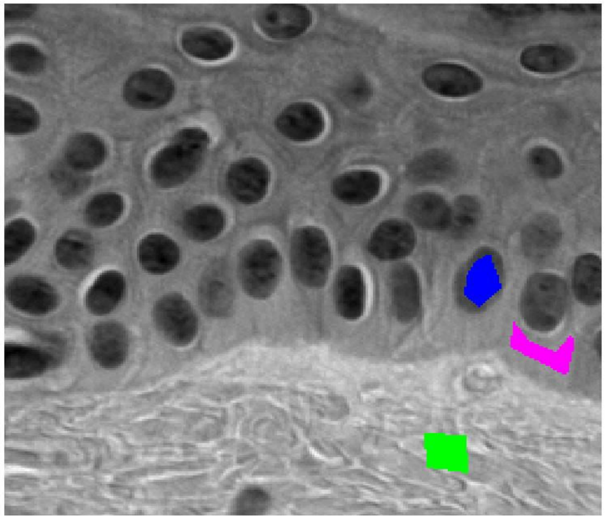
These folders correspond to segments in the data which are similar to each other at different scales of similarity.

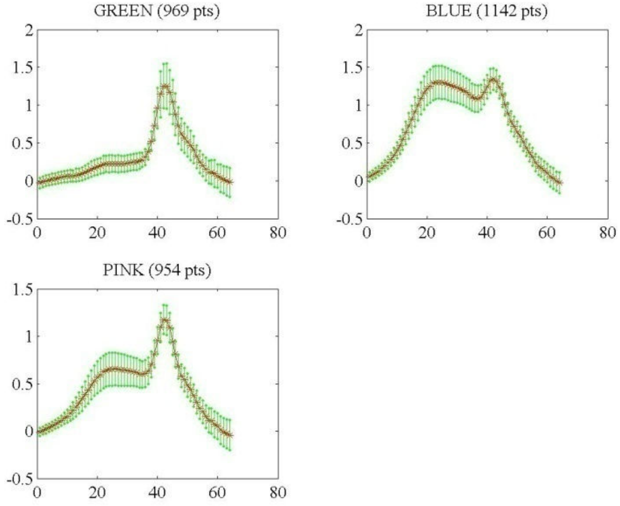
We think of a digital database, such as a hyperspectral image or a questionnaire, as a matrix whose columns as well as rows can be geometrized into their diffusion geometries, to be combined into a single robust structure.

We illustrate this approach as follows.

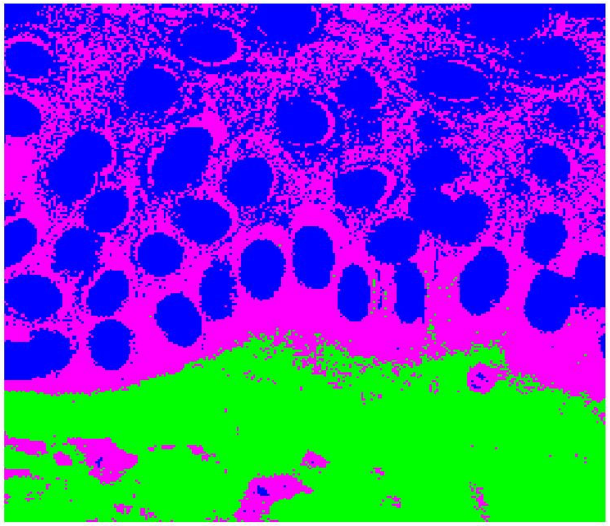
Conventional nearest neighbor search , compared with a diffusion search. The data is a pathology slide ,each pixel is a digital document (spectrum below for each class)

256 X 256 image with 861 labeled pts

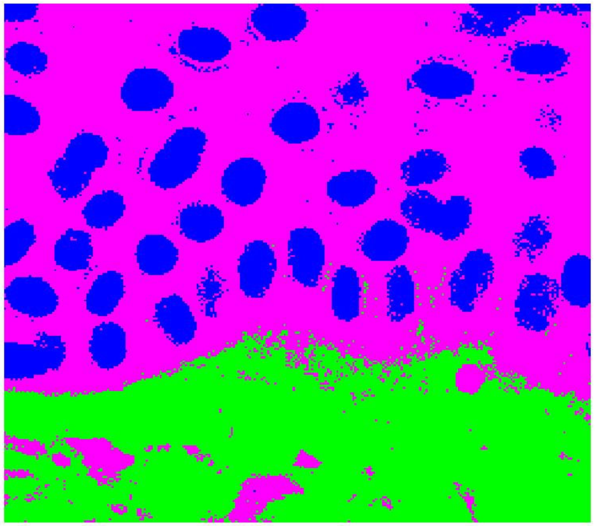




NN



Extension to all points



Our search Projection based on diffusion geometry provides the bottom image on the right

A simple empirical diffusion/inference matrix A can be constructed as follows

Let X_i ***represent normalized data (they are simply rows of a data matrix) ,we "soft truncate" the covariance matrix defining an infinitesimal affinity as***

$$A_0 = [X_i \bullet X_j]_\varepsilon = \exp\{-(1 - X_i \bullet X_j)/\varepsilon\}$$

$$\|X_i\| = 1$$

A is a renormalized Markov version of this matrix

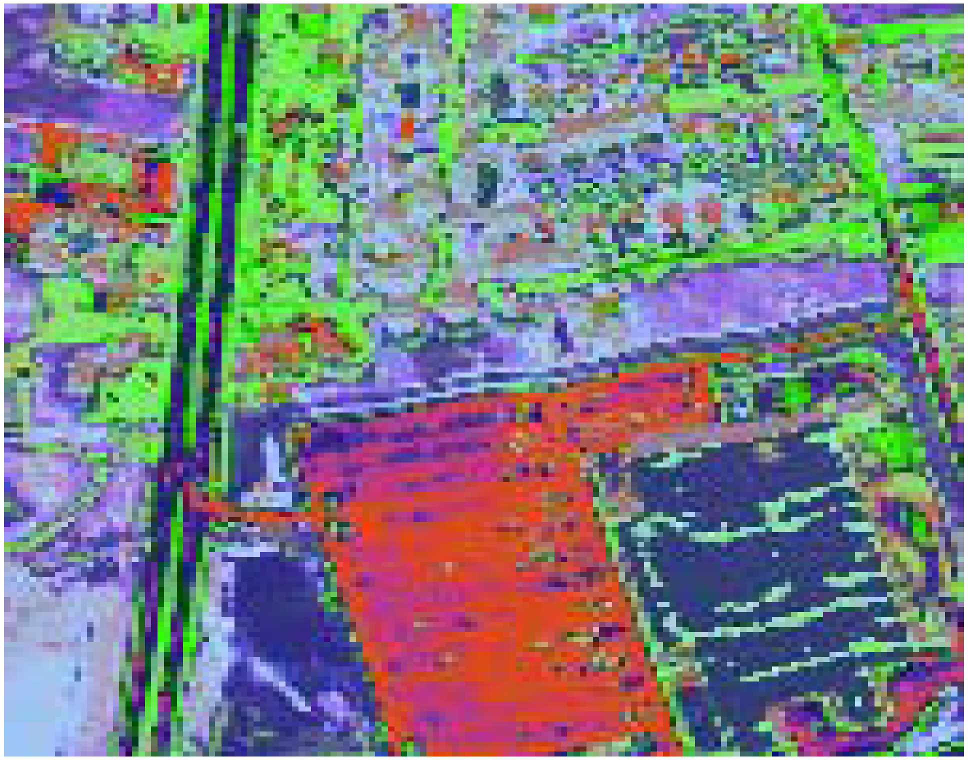
The eigenvectors of this matrix provide a local non linear principal component analysis of the data . Whose entries are the diffusion coordinates These are also the eigenfunctions of the discrete Graph Laplace Operator.

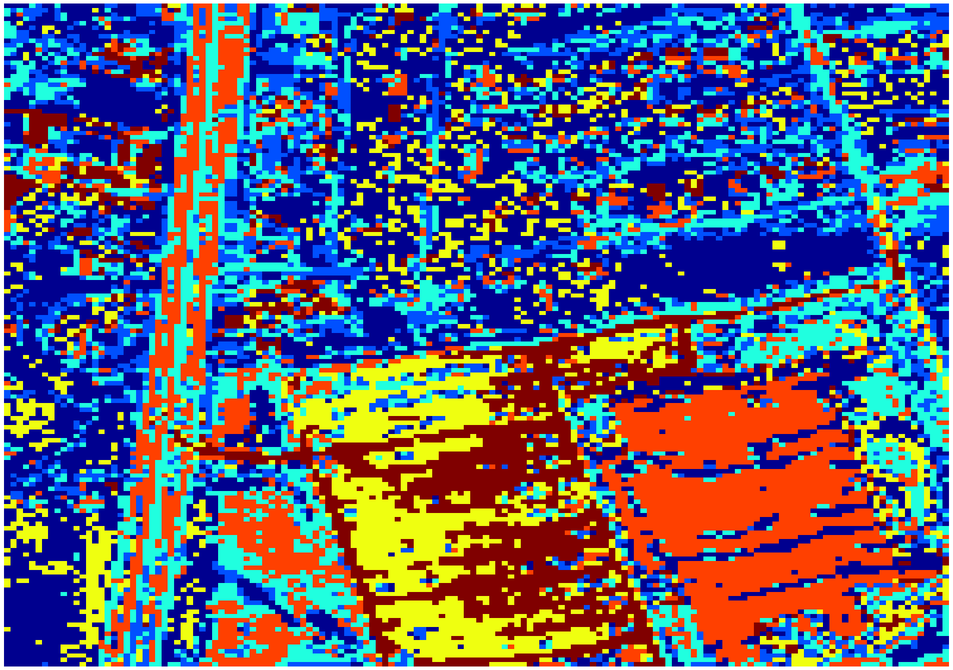
$$A^t = \sum \lambda_l^{2t} \phi_l(X_i)\phi_l(X_j) = a_t(X_i, X_j)$$

$$X_i^{(t)} \rightarrow (\lambda_1^{\,t}\phi_1(X_i), \lambda_2^{\,t}\phi_2(X_i), \lambda_3^{\,t}\phi_3(X_i), ..)$$

$$d_t^2(X_i, X_j) = a_t(X_i, X_i) + a_t(X_j, X_j) - 2a_t(X_i, X_j) = \left\| X_i^{(t)} - X_j^{(t)} \right\|^2$$

This map is a diffusion embedding into Euclidean space (at time t) .

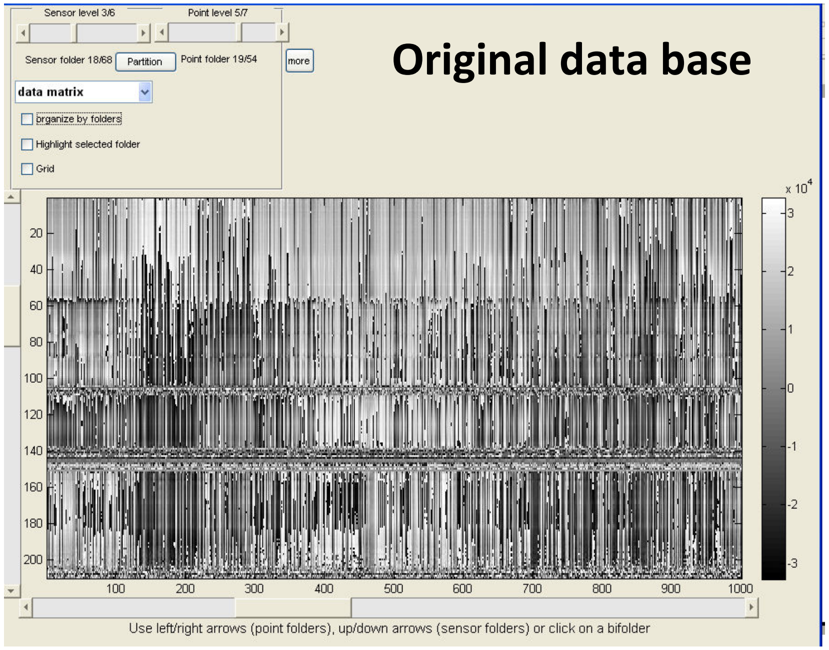




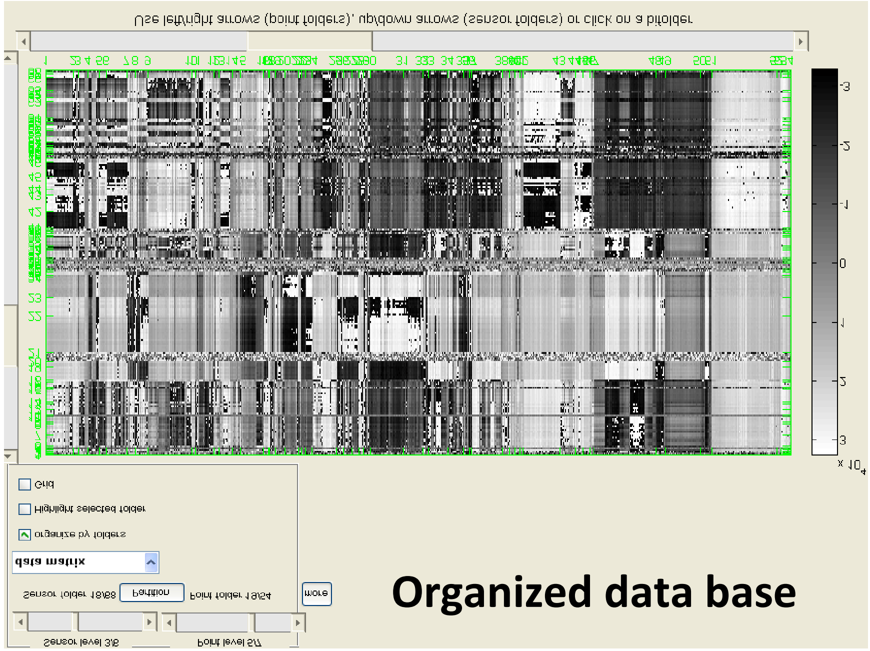
Hyperspectal image on left is
organized into a diffusion graph,
displayed in terms of 3
coordinates of embedding.
The same image is segmented
into folders of pixels whose
spectra are similar, bottom left.

We view the hyperspectral image as a questionnaire in which each pixel is interrogated spectrally, and each of the 206 spectral bands is a response. This database displayed here for a 1000 pixels is then organized as a graph of pixels using their spectral affinity as well as a graph of spectral bands, in which the affinity between bands is defined by their correlation across the data.

These two structures are decomposed into trees of folders, resulting in a joint organization of scene segments and data driven spectral folders below.

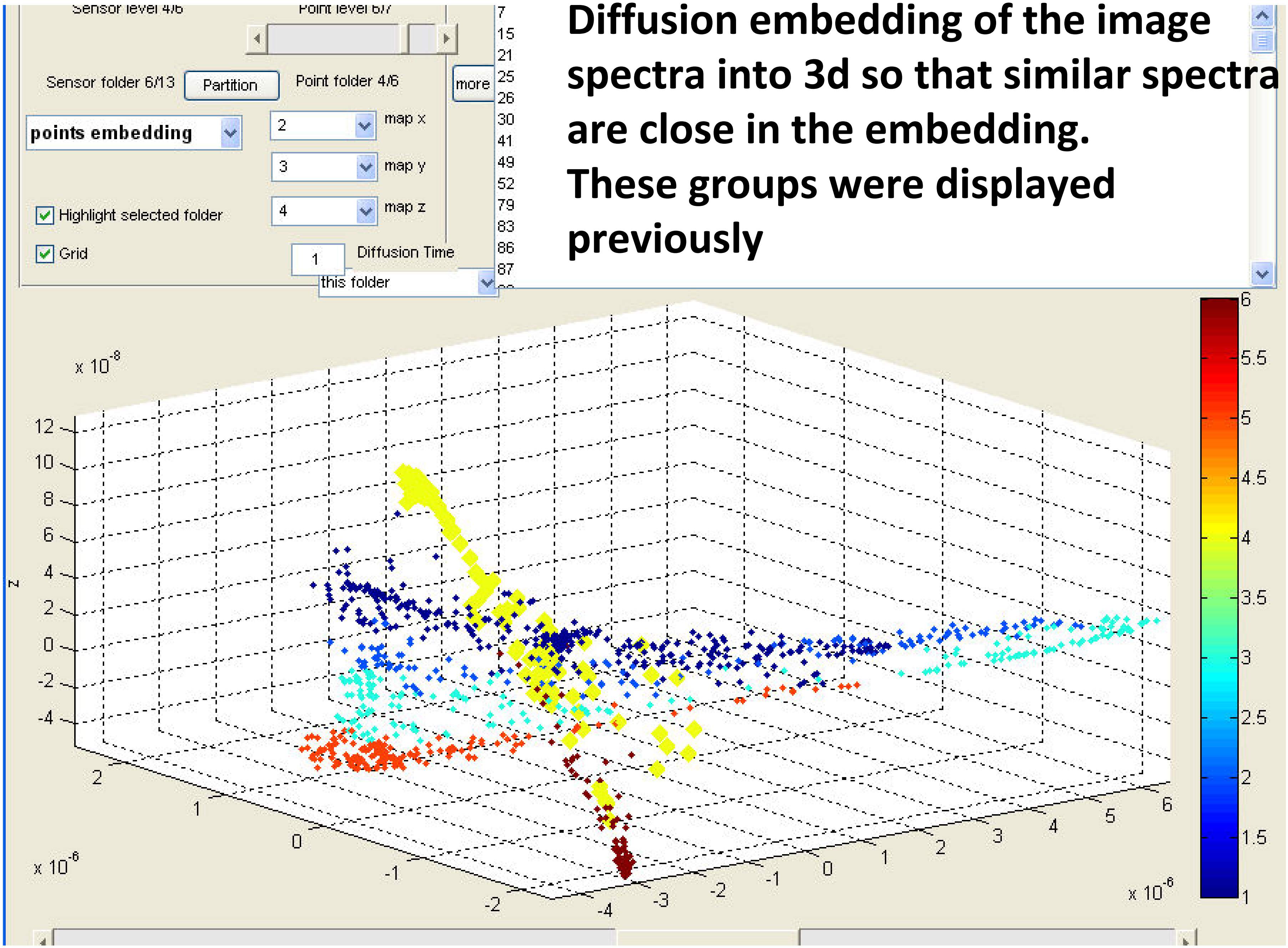

Original data base

Use left/right arrows (point folders), up/down arrows (sensor folders) or click on a bifolder


Organized data base

Diffusion embedding of the image spectra into 3d so that similar spectra are close in the embedding.

These groups were displayed previously



Sensor level 4/6

Point level 6/7

Sensor folder 6/13 Partition

Point folder 4/6

more

points embedding

2 map x

3 map y

4 map z

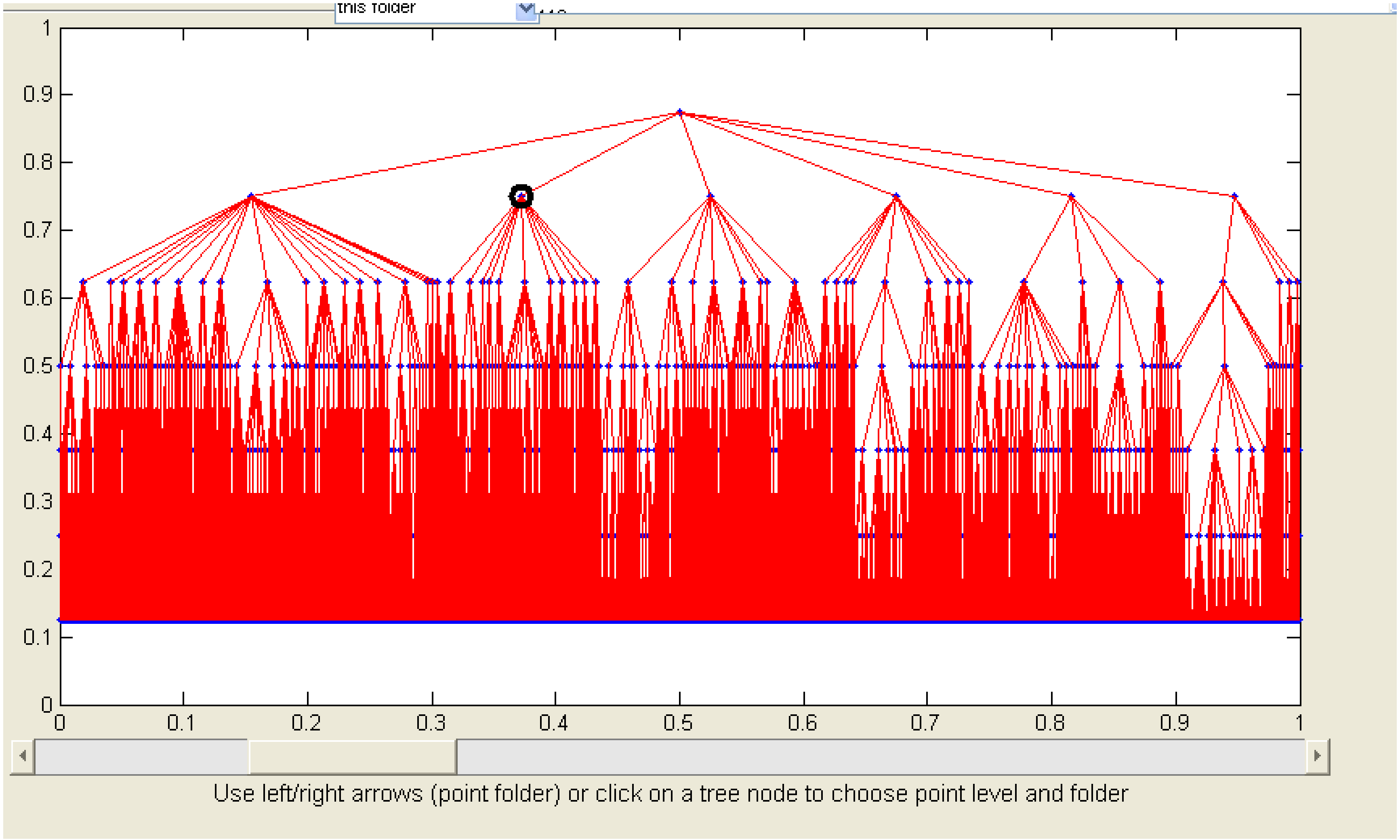
☑ Highlight selected folder

☑ Grid

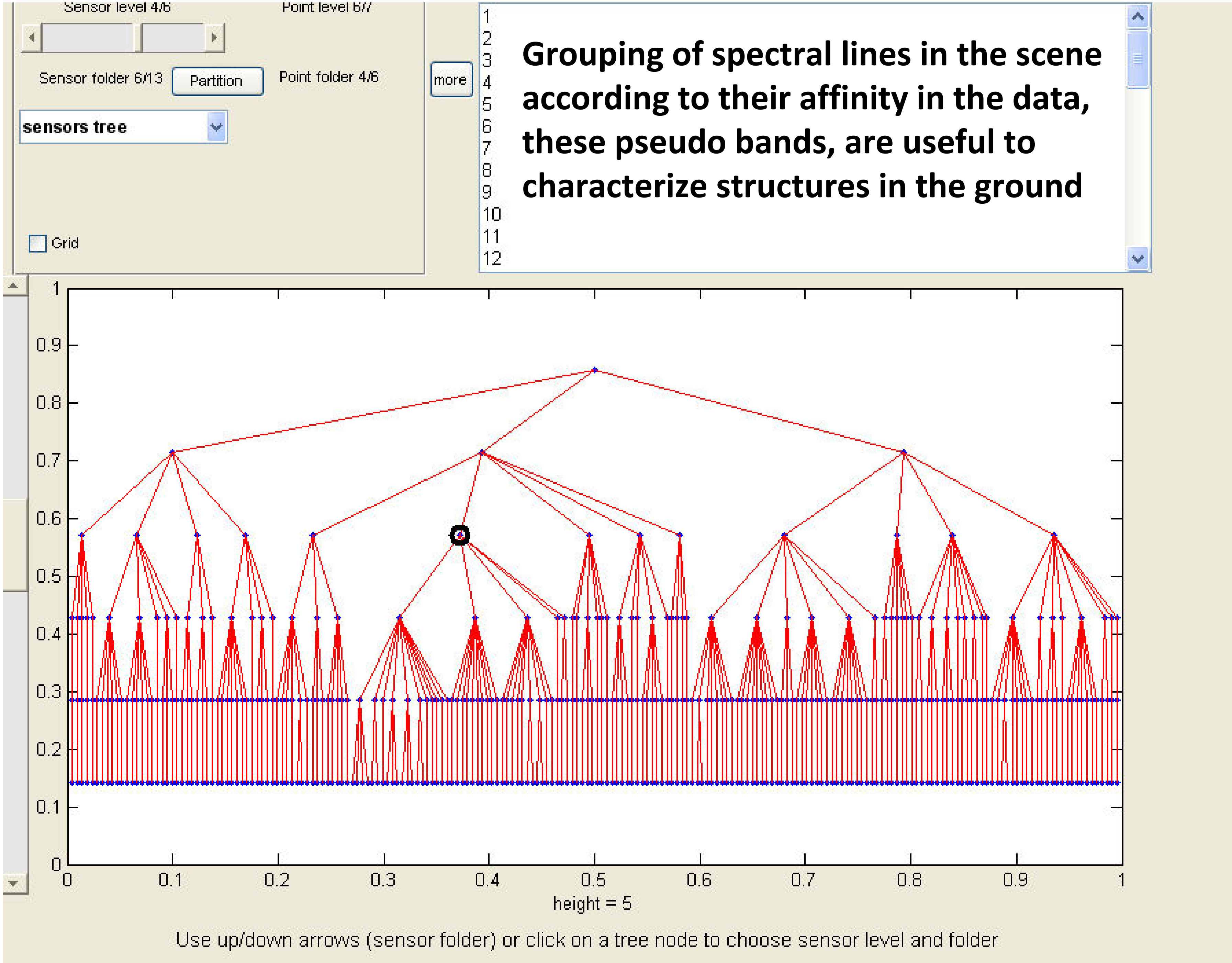
1 Diffusion Time

this folder

The tree of scene segmentation by spectral similarity displayed previously in the embedding and the image



Use left/right arrows (point folder) or click on a tree node to choose point level and folder



Sensor level 4/6

Point level 6/7

Sensor folder 6/13

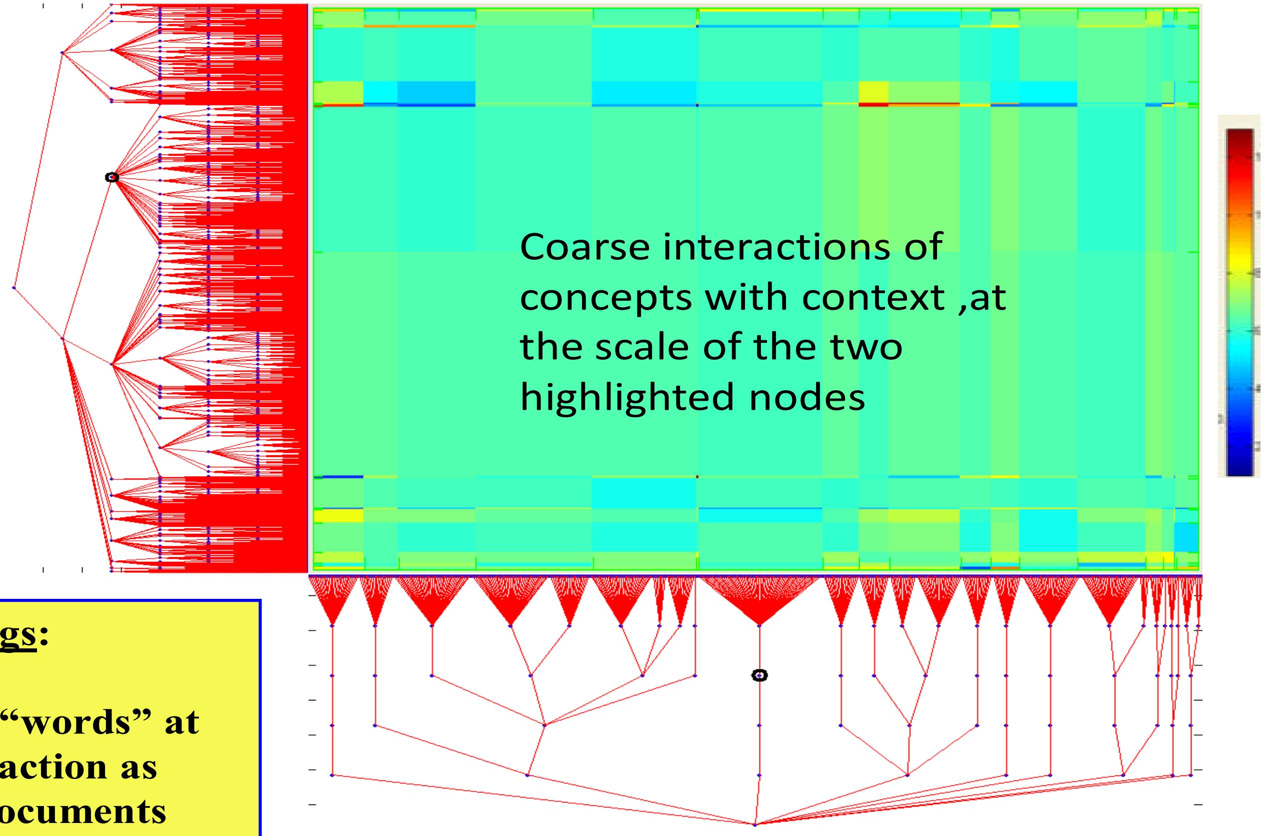
Point folder 4/6

Grouping of spectral lines in the scene according to their affinity in the data, these pseudo bands, are useful to characterize structures in the ground

Use up/down arrows (sensor folder) or click on a tree node to choose sensor level and folder

Mutual Organization / Tree Structures for context- concept duality,

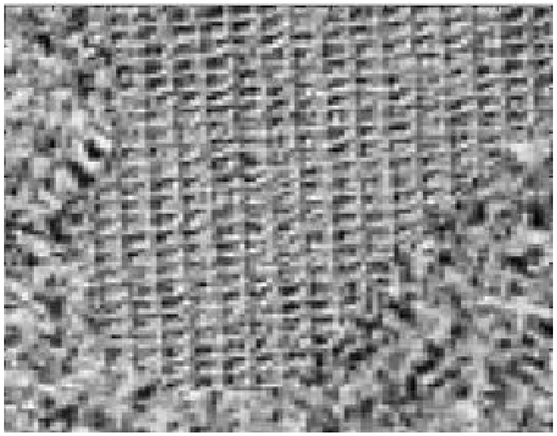
Although we use linguistic analogies these trees were built on time series of observations of 500 objects , the concepts are scenarios of times with similar responses among the population while the contexts are group of objects with similar temporal responses.

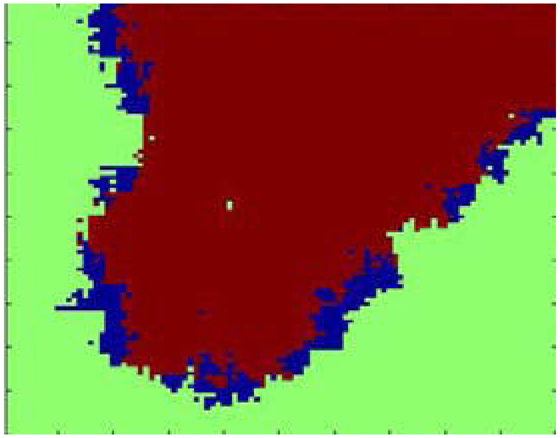


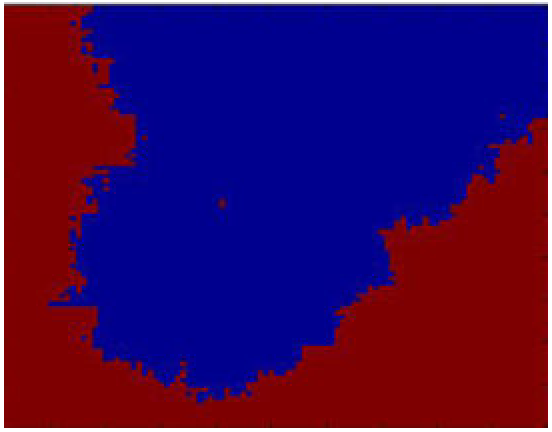
Coarse interactions of concepts with context ,at the scale of the two highlighted nodes

<u>Affinity based Groupings</u>:

Concepts - clusters of “words” at different levels of abstraction as they relate to various documents clusters =contexts
Contexts- clusters of “documents” with similar vocabulary profile







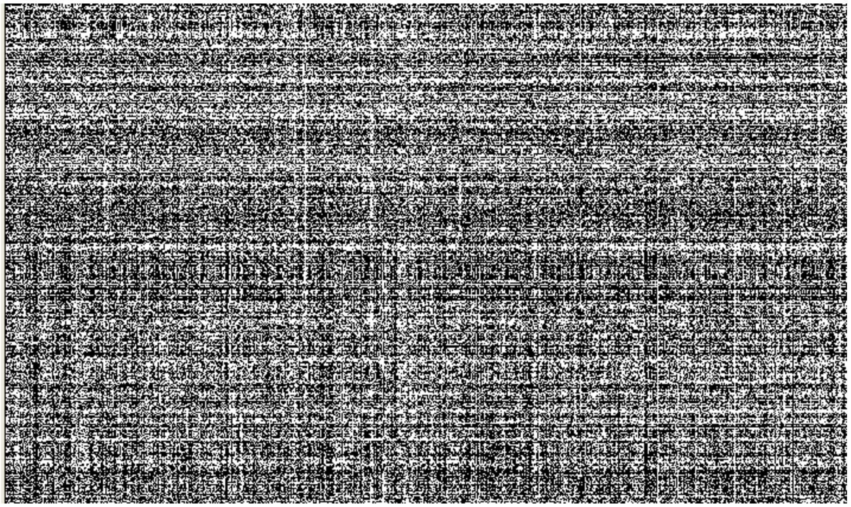
The same approach of organizing an image as a questionnaire, is effective for texture segmentation.

Here we associate with each pixel, the log values of the Fourier coefficients of the 11X11 square centered at the pixel.

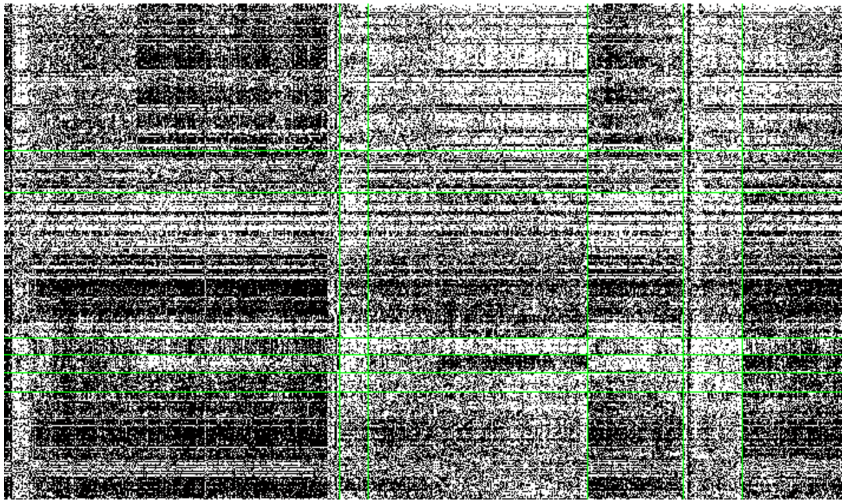
The middle image shows folders at a level before the last– observe the spot in the middle of the brown.

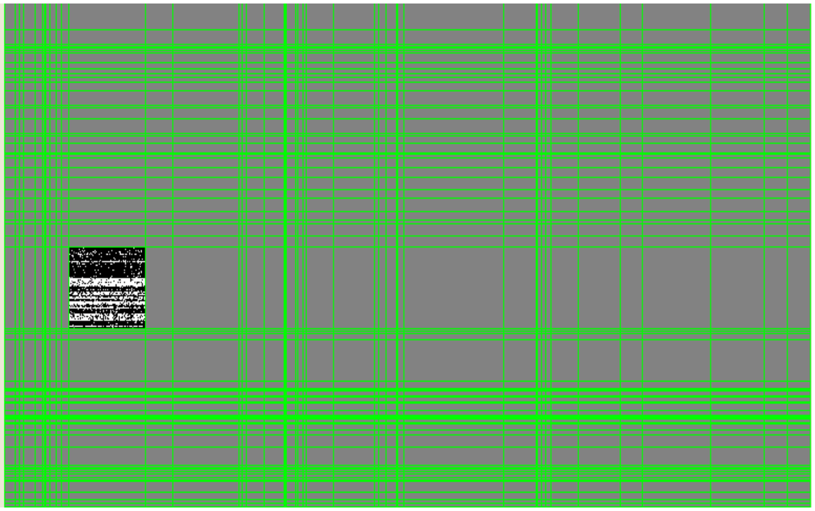
The image on the right is a good segmentation of the textures . Observe that no assumptions or filters were given. This can be done as easily without using the FT.

Conceptual analogy using psychological questionnaires, people:pixels questions:spectral band responses



A disorganized questionnaire ,on the left, the columns represent people , the row are binary questions. Mutual multiscale bi learning , organizes the data, bottom left , The questionnaire is split on a two scale grid below. Showing in the highlighted rectangle , the consistency of responses of a demographic group (context) to a group of questions (concept)





The point is that any anomalous inconsistent response is visible as noise on the bar code.

Appendix B: Technical Report: Target Detection Using Diffusion Geometry and Local Covariance Matrices

Technical Report:
Target Detection Using Diffusion Geometry and Local Covariance Matrices

1 Introduction

In this technical report we examine the target detection problem for hyperspectral imagery (HSI) data. We denote the HSI data set as

$$\mathcal{X} = \{x_{ij}\}_{i,j=1}^{l,w} \subset \mathbb{R}^D.$$

Here l and w denote the physical dimensions of the data set, while D denotes the spectral dimension. One may think of each x_{ij} as a pixel vector in $\mathbb{R}^D$ with geographic coordinates given by (i, j). We assume throughout the report that $\|x_{ij}\|_2 = 1$ for all i, j.

We shall consider targets to be small, anomalous features within the data set. By small we mean a few pixels; by anomalous we mean that the target pixels should be different than the surrounding background pixels.

2 The algorithm

The target detection algorithm can be broken into three main steps. In the first step we compute a local covariance matrix for each pixel; the second step then computes diffusion maps based on these local covariance matrices. Finally, the targets are extracted from the diffusion maps.

2.1 Local covariance matrices

Consider the pixel x_{ij}. To compute the local covariance matrix for x_{ij}, we first take a square ball around x_{ij} of radius r. By square ball of radius r we just mean a square with side length $2r+1$ centered at x_{ij}. We shall denote this ball as $\mathcal{B}_r(x_{ij})$ and formally define it as:

$$\mathcal{B}_r(x_{ij}) = \{x_{i'j'} \in \mathcal{X} : i' \in \mathbb{Z} \cap [i-r, i+r] \cap [1, l],\ \ j' \in \mathbb{Z} \cap [j-r, j+r] \cap [1, w]\}.$$

We then take a local neighborhood from within the ball $\mathcal{B}_r(x_{ij})$. This neighborhood is consists of x_{ij} plus the k closest neighbors to x_{ij} from within $\mathcal{B}_r(x_{ij})$

according to their inner product with x_{ij}. Let y_m denote the elements of $\mathcal{B}_r(x_{ij})$ sorted according to their inner product with x_{ij}, so that:

$$\langle x_{ij}, y_m \rangle \geq \langle x_{ij}, y_n \rangle \quad \text{if and only if} \quad m \leq n.$$

Note that $y_1 = x_{ij}$. We denote this neighborhood as $\mathcal{N}_{k,r}(x_{ij})$ and formally define it as:

$$\mathcal{N}_{k,r}(x_{ij}) = \{y_m \in \mathcal{B}_r(x_{ij}) : 1 \leq m \leq k+1\}.$$

We then compute a $(k+1) \times (k+1)$ local covariance matrix $C_{x_{ij}}$ from the neighborhood $\mathcal{N}_{k,r}(x_{ij})$. Let $\mu_m = \text{mean}(y_m)$; then the (m,n) entry of $C_{x_{ij}}$ is given by:

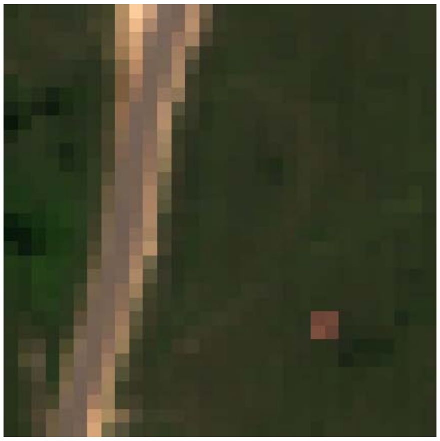
$$(C_{x_{ij}})_{mn} = \frac{1}{D-1} \sum_{p=1}^{D} (y_m(p) - \mu_m)(y_n(p) - \mu_n).$$

These local covariance matrices capture the statistics of the spectral neighbors to x_{ij} that are within a prescribed geographic radius. One can see their usefulness through the following examples.

2.1.1 Examples

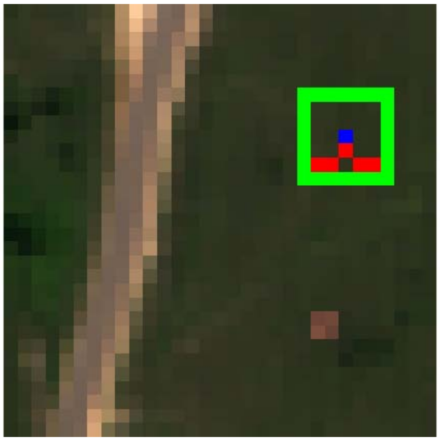
Consider figure 1, which is a patch of dimension $31 \times 31 \times 161$ depicting a road cutting through grass. A 2×2 target has also been added, which in this case is part of a house from elsewhere in the larger image.

Figure 1: Pseudo-color image of patch



In figure 2 we highlight a pixel taken from the grass (highlighted in blue), and depict the boundary of its square ball of radius $r = 2$ (highlighted in green) as well as its $k = 5$ closest neighbors (highlighted in red). The associated local

Figure 2: Grass pixel and neighbors



covariance matrix is given by:

$$C_{grass} = \begin{pmatrix} 0.0063 & 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0062 \\ 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0062 \\ 0.0062 & 0.0062 & 0.0063 & 0.0062 & 0.0062 & 0.0062 \\ 0.0062 & 0.0062 & 0.0062 & 0.0063 & 0.0062 & 0.0062 \\ 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0062 \\ 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0062 \end{pmatrix}$$

We note that the local neighbors of this are also grass pixels, and as such have similar spectral signatures. Thus the local covariance matrix is nearly a constant (in this case 1/160) times a matrix of ones.

Now consider a second pixel, this one taken from the road; see figure 3. Notice that the $k = 5$ closest neighbors are all taken from the road as well, and that none come from the grass. This implies that the local covariance matrix of the road pixel should be similar to the local covariance matrix of the grass pixel. Indeed, the local covariance matrix of the road pixel is:

$$C_{road} = \begin{pmatrix} 0.0063 & 0.0062 & 0.0062 & 0.0062 & 0.0062 & 0.0061 \\ 0.0062 & 0.0063 & 0.0062 & 0.0062 & 0.0061 & 0.0061 \\ 0.0062 & 0.0062 & 0.0062 & 0.0061 & 0.0060 & 0.0062 \\ 0.0062 & 0.0062 & 0.0061 & 0.0063 & 0.0062 & 0.0059 \\ 0.0062 & 0.0061 & 0.0060 & 0.0062 & 0.0063 & 0.0058 \\ 0.0061 & 0.0061 & 0.0062 & 0.0059 & 0.0058 & 0.0063 \end{pmatrix}$$

Finally, we consider a pixel taken from the target house; see figure 4. Unlike the previous two examples, since the target is both anomalous and small, some of the neighbors of the target pixel must in fact not be spectrally similar. In

Figure 3: Road pixel and neighbors

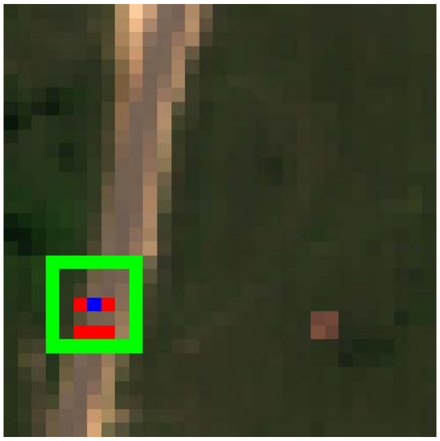
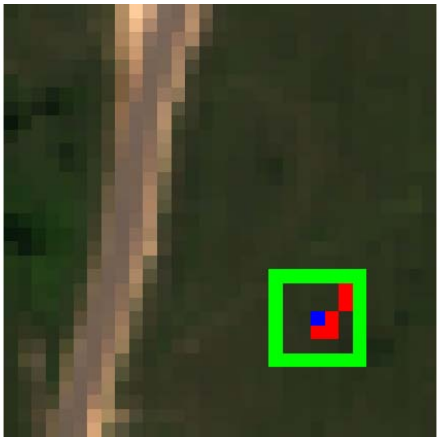


Figure 4: Target pixel and neighbors



this case, two neighbors of the house pixel are grass pixels. As such, we would expect the local covariance matrix to indicate this; indeed, we have:

$$C_{target} = \begin{pmatrix} 0.0062 & 0.0062 & 0.0062 & 0.0061 & 0.0042 & 0.0042 \\ 0.0062 & 0.0062 & 0.0062 & 0.0061 & 0.0042 & 0.0042 \\ 0.0062 & 0.0062 & 0.0063 & 0.0059 & 0.0042 & 0.0041 \\ 0.0061 & 0.0061 & 0.0059 & 0.0063 & 0.0041 & 0.0042 \\ 0.0042 & 0.0042 & 0.0042 & 0.0041 & 0.0063 & 0.0062 \\ 0.0042 & 0.0042 & 0.0041 & 0.0042 & 0.0063 & 0.0062 \end{pmatrix}$$

Computing the Frobenius norm (denoted $\| \cdot \|_F$) of the differences between the local covariance matrices, we get:

$$\begin{aligned} \|C_{grass} - C_{road}\|_F &= 0.00092 \\ \|C_{grass} - C_{target}\|_F &= 0.0083 \\ \|C_{road} - C_{target}\|_F &= 0.0078 \end{aligned}$$

Comparing, we see that the norm of the difference between the two non-target (background) pixels is an order of magnitude smaller than the norm of the difference between a target pixel and a background pixel.

2.2 Diffusion maps

Using the local covariance matrices and the Frobenius norm we define the following distance on the data set $\mathcal{X}$:

$$d(x_{ij}, x_{i'j'}) = \|C_{x_{ij}} - C_{x_{i'j'}}\|_F.$$

As illustrated above, this distance has the property that the distance between two background pixels will be small, whereas the distance between a background pixel and a target pixel will be large. Using this distance, we define the following kernel for the data set $\mathcal{X}$:

$$k(x_{ij}, x_{i'j'}) = e^{-\|C_{x_{ij}} - C_{x_{i'j'}}\|_F / \varepsilon}, \quad \varepsilon > 0.$$

Now set

$$\omega(x_{ij}) = \sum_{x_{i'j'} \in \mathcal{X}} k(x_{i,j}, x_{i'j'}),$$

and then define the normalized kernel p as:

$$p(x_{ij}, x_{i'j'}) = \frac{k(x_{ij}, x_{i'j'})}{\omega(x_{ij})}.$$

We now reorder the elements of $\mathcal{X}$ so that they are in list form, and denote this reordering as (slightly abusing notation):

$$\mathcal{X} = \{x_i\}_{i=1}^N, \quad N = l \cdot w.$$

Define the $N \times N$ matrix P as:

$$P_{ij} = p(x_i, x_j).$$

We compute the eigenvectors and eigenvalues of P, which we denote as $\{\psi_i\}_{i=0}^{N-1}$ and $\{\lambda_i\}_{i=0}^{N-1}$, respectively. Note that ψ_0 is constant and that $1 = \lambda_0 \geq |\lambda_1| \geq |\lambda_2| \geq \ldots$. The diffusion mapping is then given by:

$$x_i \mapsto \Psi(x_i) = (\lambda_1 \psi_1(i), \ldots, \lambda_s \psi_s(i)) \in \mathbb{R}^s.$$

In the above line s is defined to be the unique value such that:

$$\frac{\sum_{i=1}^{s-1} |\lambda_i|}{\sum_{i=1}^{N-1} |\lambda_i|} < \delta \leq \frac{\sum_{i=1}^{s} |\lambda_i|}{\sum_{i=1}^{N-1} |\lambda_i|},$$

where $0 \leq \delta \leq 1$ is an accuracy parameter. Note that the eigenvalues will decay very fast due to the fact that most pixels x_i and x_j will satisfy $k(x_i, x_j) \approx 1$. Thus taking a large δ would allow one to retain most of the information in the diffusion embedding, but would still result in a small value of s. Furthermore, due to the fast decay of the eigenvalues, in practice one need not compute all of them, but rather a large enough amount so that the value of s is computed accurately.

2.2.1 Example

We now return to the data set from figure 1. Setting $r = 2$, $k = 5$, $\varepsilon = 1$, and $\delta = 0.99$, we compute the local covariance diffusion map for this data set. Using these parameters and the definition of s, we kept $s = 7$ eigenvectors. A scatter plot of $\lambda_1 \psi_1$ versus $\lambda_2 \psi_2$ is given in figure 5. The red circles correspond to the target pixels, while the blue circles correspond to the remaining pixels. Figures 6 and 7 show the first two eigenvectors.

2.3 Extracting the targets

We now extract the targets based on the diffusion coordinates. As exhibited in figure 5, the diffusion coordinates of the target pixels should be 'far away' from the diffusion coordinates of the background pixels. Rather than computing all pairwise diffusion distances though, we note that the diffusion norm of the background pixels is small relative to the diffusion norm of the target pixels. Therefore, to extract the targets, we examine the norm of each new diffusion map, that is $\|\Psi(x_i)\|_2$.

2.3.1 Example

We return once more to the data set from figure 1. Using the diffusion maps computed in section 2.2.1, we compute the norm of the diffusion map for each pixel. The results are given in figure 8. After suitable thresholding, the lower size norms fall out and only the targets remain; see figure 9.

Figure 5: Scatter plot of first two eigenvectors

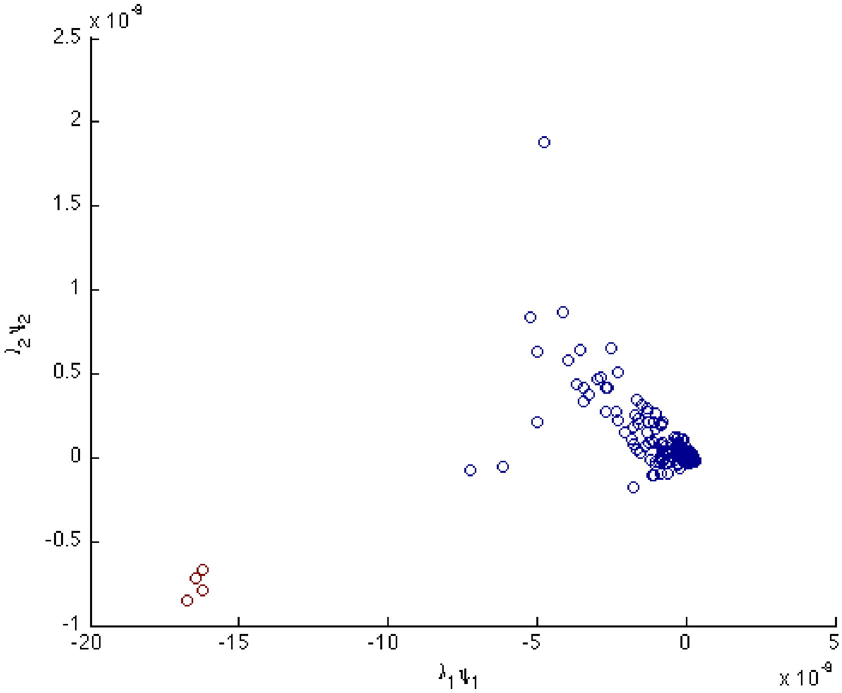


Figure 6: First eigenvector

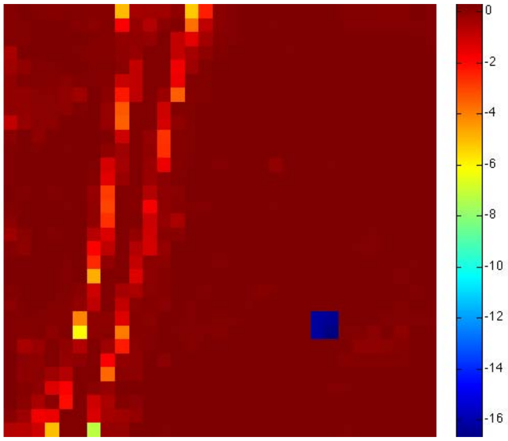


Figure 7: Second eigenvector

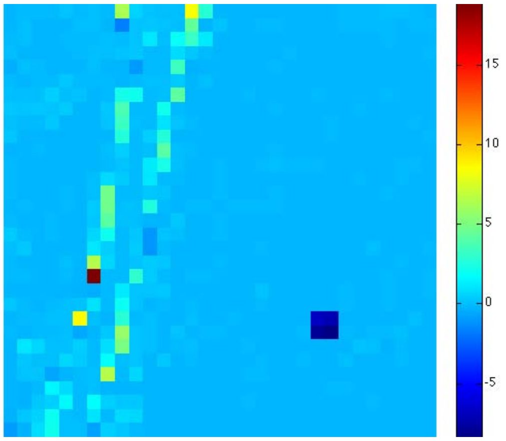


Figure 8: Diffusion norms

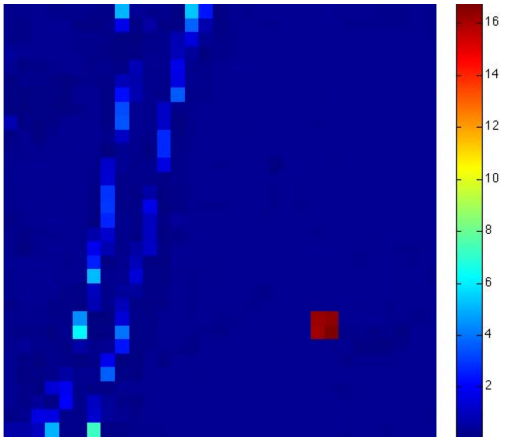
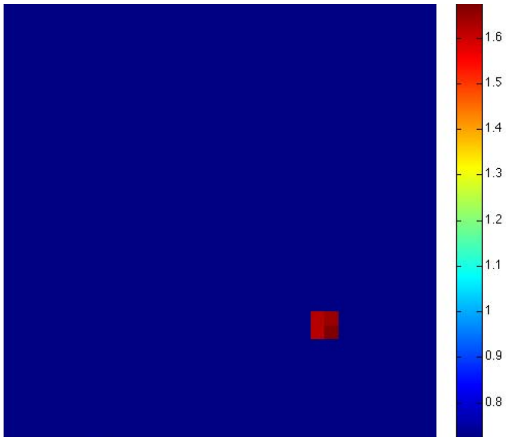


Figure 9: Thresholded diffusion norms



3 Experiments

We now present some further experiments.

3.1 Multiple targets

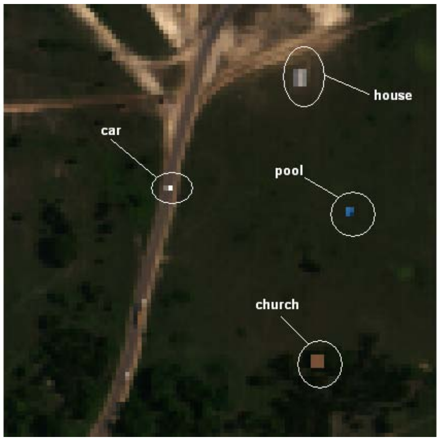
We consider a $100 \times 100 \times 161$ patch to which we have added 4 targets of various sizes and materials; see figure 10 for a pseudo-color image. The four targets and their dimensions are:

1. car, 1×2
2. part of a church, 3×3
3. pool, 2×2
4. house, 4×3

We ran the local covariance diffusion maps algorithm with the following settings: $r = 3$, $k = 15$, $\varepsilon = 1$, and $\delta = 0.8$. The number of eigenvectors retained was $s = 4$. A scatter plot of the first two eigenvectors is shown in figure 11; the colors represent the following pixels:

1. dark blue - background pixels
2. light blue - car
3. green - church
4. orange - pool
5. red - house

Figure 10: Pseudo-color image



The four eigenvectors are shown in figure 12. The norms of each pixel's diffusion map are given in figure 13; a thresholded version is given in figure 14.

3.1.1 Evaluation of results

Examining figures 13 and 14 more closely, we make the following points:

- In figure 13 each target pixel is highlighted.
- Also in figure 13 it appears that there may be some other cars on the road that were part of the original image.
- The brightest pixels are shown in figure 14. In this figure, we see that at least one pixel from each target is highlighted. More specifically, the entire house is highlighted, as well as the entire church; 3/4 pool pixels are visible, and 1/2 car pixels are visible.
- A closer examination of the two missing pixels in figure 14 reveals that they are most likely mixed pixels, which could explain their lower intensity.
- There is one faint false positive pixel in figure 14.

4 Possible Extensions

4.1 Generalized covariance matrices

Suppose that we normalize the data set $\mathcal{X}$ such that not only $\|x_{ij}\|_2 = 1$ but also mean$(x_{ij}) = 0$. The local covariance matrix for x_{ij} would then be:

$$(C_{x_{ij}})_{mn} = \frac{1}{D-1}\sum_{p=1}^{D} y_m(p)y_n(p) = \frac{1}{D-1}\langle y_m, y_n\rangle.$$

Figure 11: Scatter plot of first two eigenvectors

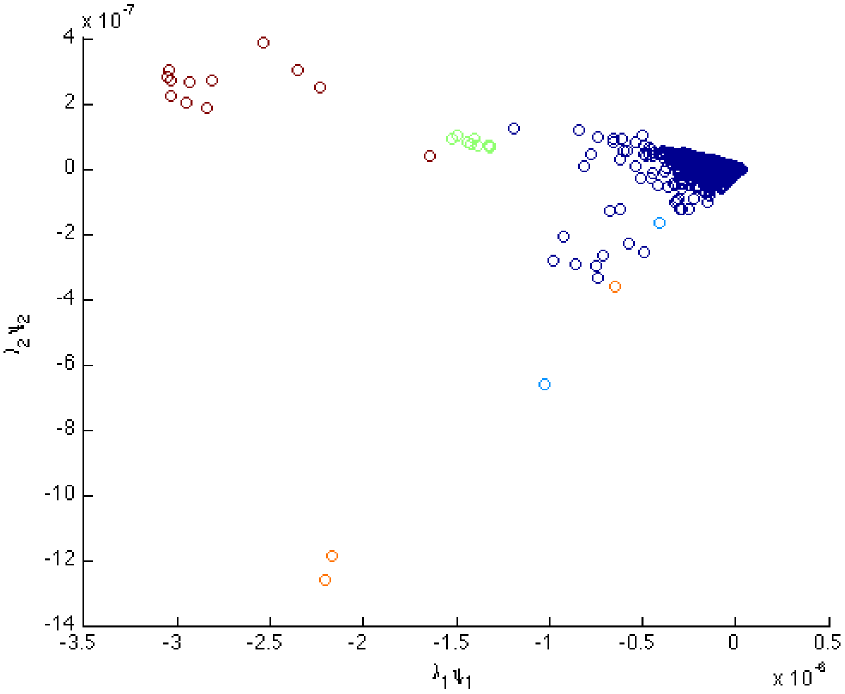
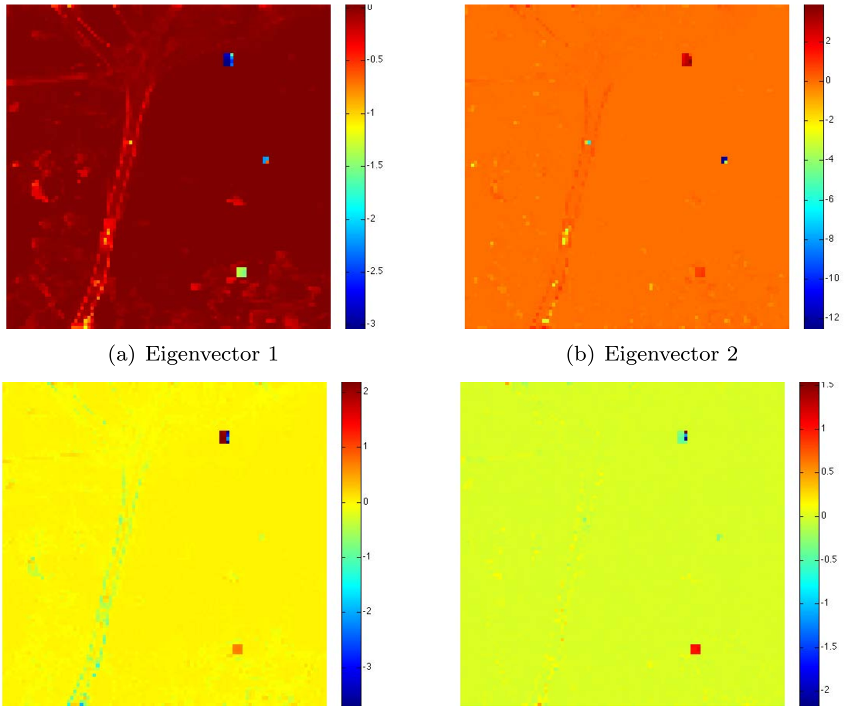


Figure 12: Eigenvectors



(a) Eigenvector 1

(b) Eigenvector 2

(c) Eigenvector 3

(d) Eigenvector 4

Figure 13: Diffusion norms

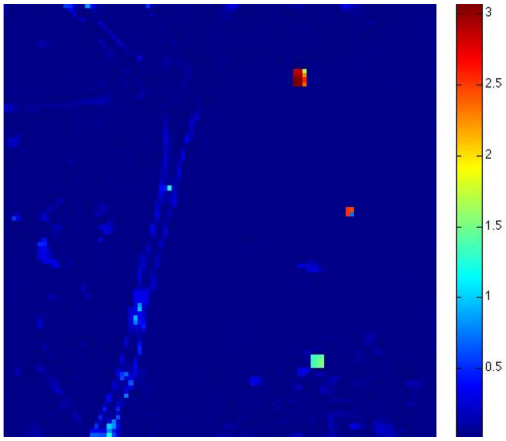
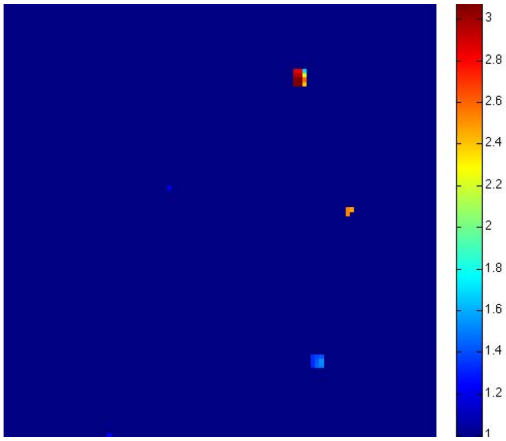


Figure 14: Thresholded diffusion norms



Thus we are using the inner product as a measure of the similarity between x_{ij} and the pixels y_m in its local neighborhood. We could perhaps improve the results by using a more sophisticated measure of similarity, which we denote by the general function f. Thus we have a generalized covariance matrix:

$$(C_{x_{ij}})_{mn} = f(y_m, y_n).$$

Appendix C: Technical Report: Change Detection Using Diffusion Geometry

Technical Report: Change Detection Using Diffusion Geometry

1 Introduction

In this technical report we examine the change detection problem for hyperspectral imagery (HSI) data. We assume that we have two hyperspectral data sets $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$ which depict the same scene, but were taken at different times:

$$\mathcal{X}^{(1)} = \{x_{ij}^{(1)}\}_{i,j=1}^{l,w} \subset \mathbb{R}^D.$$

$$\mathcal{X}^{(2)} = \{x_{ij}^{(2)}\}_{i,j=1}^{l,w} \subset \mathbb{R}^D.$$

Here l and w denote the physical dimensions of the data set, while D denotes the spectral dimension. One may think of each $x_{ij}^{(\alpha)}$ as a pixel vector in $\mathbb{R}^D$ with geographic coordinates given by (i,j). These coordinates are collected in the set $\mathcal{I}$, i.e.

$$\mathcal{I} = \{(i,j) : i \in \mathbb{Z} \cap [1,l], \quad j \in \mathbb{Z} \cap [1,w]\}.$$

The idea behind change detection is to determine the anomalous changes that occur from day one to day two. Atmospheric changes, changes in the lighting, and other changes that affect the entire area should not be taken into account.

1.1 Data normalization

Without some sort of data normalization, accurately detecting changes between the two days is nearly impossible. To that end we normalize both data sets so that:

$$\|x_{ij}^{(\alpha)}\|_2 = 1 \quad \text{and} \quad \text{mean}(x_{ij}^{(\alpha)}) = 0.$$

Note that normalizing the norm so that it equals one for each data point seems to be the key part. Setting the mean of each vector to zero seems to have less influence on the final outcome; perhaps it is not necessary or something else is more appropriate (such as subtracting out the mean of the data set).

2 Simple solutions

We present some possible simple solutions.

2.1 Inner product

One can use the inner product as a measure of similarity, i.e.:

$$d_{ip}(x_{ij}^{(1)}, x_{ij}^{(2)}) = \arccos(\langle x_{ij}^{(1)}, x_{ij}^{(2)} \rangle). \tag{2.1}$$

See section 6 for some examples.

2.2 ℓ^2 distance

One can use the ℓ^2 distance as a measure of similarity, i.e.:

$$d_{\ell^2}(x_{ij}^{(1)}, x_{ij}^{(2)}) = \|x_{ij}^{(1)} - x_{ij}^{(2)}\|_2 \tag{2.2}$$

See section 6 for some examples.

2.3 Total ℓ^2 distance

One can also compare the distances within one day to the distances within the second day. More specifically, we compute:

$$d_{\text{total } \ell^2}(x_{ij}^{(1)}, x_{ij}^{(2)}) = \left(\sum_{(i',j') \in \mathcal{I}} \left[d_{\ell^2}(x_{ij}^{(1)}, x_{i'j'}^{(1)}) - d_{\ell^2}(x_{ij}^{(2)}, x_{i'j'}^{(2)}) \right]^2 \right)^{\frac{1}{2}} \tag{2.3}$$

See section 6 for some examples.

2.4 PCA ℓ^2 distance

One can first compute the PCA mapping of each day, normalizing so that each new dimension has variance one; denote this mapping as:

$$\mathcal{X}_{pca}^{(\alpha)} = \{\tilde{x}_{ij}^{(\alpha)} : (i,j) \in \mathcal{I}\} \subset \mathbb{R}^d.$$

Note that the dimension d can either be selected or one can set it as the dimension required to retain a certain percentage of the energy in the data set. The distance between pixels is then given by:

$$d_{\text{PCA } \ell^2}(x_{ij}^{(1)}, x_{ij}^{(2)}) = \|\tilde{x}_{ij}^{(1)} - \tilde{x}_{ij}^{(2)}\|_2. \tag{2.4}$$

3 Local coordinates for change detection

We compute new local coordinates for each data set. To do this we first take a square ball around $x_{ij}^{(\alpha)}$ of radius r. By square ball of radius r we just mean a square with side length $2r+1$ centered at $x_{ij}^{(\alpha)}$. We shall denote this ball as $\mathcal{B}_r(x_{ij}^{(\alpha)})$ and formally define it as:

$$\mathcal{B}_r(x_{ij}^{(\alpha)}) = \{x_{i'j'}^{(\alpha)} \in \mathcal{X}^{(\alpha)} : (i',j') \in \mathcal{I} \cap [i-r, i+r] \times [j-r, j+r]\}.$$

We then compute PCA coordinates for each ball, and retain 99% of the energy (the 99% number can be changed of course); let d_{ij} be the dimension required by PCA to retain this amount of the energy. We set $\{z_{i'j'}^{(\alpha)}\} \subset \mathbb{R}^d$ to denote the PCA coordinates of $\mathcal{B}_r(x_{ij}^{(\alpha)})$, and let $\mu_1^{(\alpha)}, \ldots, \mu_{d_{ij}}^{(\alpha)}$ be the corresponding eigenvalues of each PCA dimension. Before proceeding we normalize the PCA coordinates so that each dimension has variance equal to one; that is we compute:

$$y_{i'j'}^{(\alpha)} = \left(\frac{1}{\sqrt{\mu_1^{(\alpha)}}} z_{i'j'}^{(\alpha)}(1), \ldots, \frac{1}{\sqrt{\mu_{d_{ij}}^{(\alpha)}}} z_{i'j'}^{(\alpha)}(d_{ij}) \right).$$

We now mark the coordinates of the pixels in $\mathcal{B}_r(x_{ij}^{(\alpha)})$ that are spectrally close to x_{ij} in the PCA coordinates. Set $r_{ij}^{(\alpha)} = \|y_{ij}^{(\alpha)}\|_2$; we compute:

$$\mathcal{I}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r) = \{(i', j') : \|y_{ij}^{(\alpha)} - y_{i'j'}^{(\alpha)}\|_2 \leq r_{ij}^{(\alpha)}\}.$$

Using these coordinates we form a spectral neighborhood of $x_{ij}^{(\alpha)}$ from within the geographic ball $\mathcal{B}_r(x_{ij}^{(\alpha)})$:

$$\mathcal{N}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r) = \{x_{i'j'}^{(\alpha)} \in \mathcal{B}_r(x_{ij}^{(\alpha)}) : (i', j') \in \mathcal{I}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r)\}.$$

We now use this local neighborhood to compute three representative of $x_{ij}^{(\alpha)}$ which we will later use to compute graphs. The first of these is the average vector of $\mathcal{N}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r)$:

$$\bar{x}_{ij}^{(\alpha)} = \frac{1}{\left|\mathcal{I}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r)\right|} \left(\sum_{(i',j') \in \mathcal{I}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r)} x_{i'j'}^{(\alpha)} \right). \tag{3.1}$$

To compute the other two, we first compute the $D \times D$ covariance matrix of $\mathcal{N}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r)$, which we shall denote $C_{ij}^{(\alpha)}$. Let $k = |\mathcal{I}_{r_{ij}^{(\alpha)}}(x_{ij}^{(\alpha)}; r)|$, and note that if $k < D$ (which is usual), then the rank of $C_{ij}^{(\alpha)}$ is k. We then compute the k nonzero eigenvalues of $C_{ij}^{(\alpha)}$ and their corresponding eigenvectors; we denote these two sets, respectively, as follows:

$$\Lambda_r(x_{ij}^{(\alpha)}) = \{\lambda_1(x_{ij}^{(\alpha)}), \ldots, \lambda_k(x_{ij}^{(\alpha)})\}, \tag{3.2}$$

$$\mathcal{V}_r(x_{ij}^{(\alpha)}) = \{v_1[x_{ij}^{(\alpha)}], \ldots, v_k[x_{ij}^{(\alpha)}]\}. \tag{3.3}$$

Combining equations (3.1), (3.2), and (3.3), we have the following map on the set $\mathcal{X}^{(\alpha)}$:

$$x_{ij}^{(\alpha)} \mapsto \left(\bar{x}_{ij}^{(\alpha)}, \, \Lambda_r(x_{ij}^{(\alpha)}), \, \mathcal{V}_r(x_{ij}^{(\alpha)}) \right). \tag{3.4}$$

3.1 Constructing the graphs and comparing them

Given the new coordinates defined in (3.4), we now use them to construct graphs pertaining to $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$. The weights of these graphs are given by the following:

$$\eta_r(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)}) = \left(\sum_{l=1}^{k} \frac{1}{\lambda_l(x_{ij}^{(\alpha)})} \left| \langle \bar{x}_{ij}^{(\alpha)} - \bar{x}_{i'j'}^{(\alpha)}, v_l[x_{ij}^{(\alpha)}] \rangle \right|^2 \right)^{\frac{1}{2}} \tag{3.5}$$

Taking $\alpha = 1$ and $\alpha = 2$ will give us two graphs, one for day one and one for day two. We compare these two graphs in order to gain a similarity measure between $x_{ij}^{(1)}$ and $x_{ij}^{(2)}$. This similarity measure is defined as:

$$d_{\text{total }\eta_r}(x_{ij}^{(1)}, x_{ij}^{(2)}) = \left(\sum_{(i',j') \in \mathcal{I}} \left[\eta_r(x_{ij}^{(1)}, x_{i'j'}^{(1)}) - \eta_r(x_{ij}^{(2)}, x_{i'j'}^{(2)}) \right]^2 \right)^{\frac{1}{2}} \tag{3.6}$$

We can then plot the entries of $d_{\text{total }\eta_r}$ to get a two dimensional map depicting which pixels changed and which did not. See section 6 for examples. Note that the $d_{\text{total }\eta_r}$ map is like the $d_{\text{total }\ell^2}$ map, except that it is computed with these new coordinates and with the distance defined by (3.5).

3.2 Diffusion maps

We can take the previous section one step further by computing diffusion maps based on the distance defined in (3.5). More specifically, define the following kernel:

$$k_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)}) = e^{-\eta_r(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2/\varepsilon}, \quad \varepsilon > 0. \tag{3.7}$$

Now set

$$\omega_{r,\varepsilon}(x_{ij}^{(\alpha)}) = \sum_{(i',j') \in \mathcal{I}} k_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)}),$$

and then define the normalized kernel $p_{r,\varepsilon}$ as:

$$p_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)}) = \frac{k_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})}{\omega_{r,\varepsilon}(x_{ij}^{(\alpha)})}. \tag{3.8}$$

Let $P_{r,\varepsilon}^{(\alpha)}$ be the $N \times N$ (where $N = l \cdot w$) matrix associated with $p_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})$. We compute the eigenvectors and eigenvalues of $P_{r,\varepsilon}^{(\alpha)}$, which we denote as $\{\psi_i^{(\alpha)}\}_{i=0}^{N-1}$ and $\{\xi_i^{(\alpha)}\}_{i=0}^{N-1}$, respectively. Note that $\psi_0^{(\alpha)}$ is constant and that $1 = \xi_0^{(\alpha)} \geq |\xi_1^{(\alpha)}| \geq |\xi_2^{(\alpha)}| \geq \ldots$. The s-dimensional diffusion mapping at time t of day α is then given by:

$$x_{ij}^{(\alpha)} \mapsto \Psi_{r,\varepsilon,t}^{(\alpha)}(x_{ij}^{(\alpha)}) = \left(\left(\xi_1^{(\alpha)}\right)^t \psi_1^{(\alpha)}(x_{ij}^{(\alpha)}), \ldots, \left(\xi_s^{(\alpha)}\right)^t \psi_s^{(\alpha)}(x_{ij}^{(\alpha)}) \right) \in \mathbb{R}^s.$$

We can now detect change by examining the following map:

$$\rho_{r,\varepsilon,t}(x_{ij}^{(1)}, x_{ij}^{(2)}) = \|\Psi_{r,\varepsilon,t}^{(1)}(x_{ij}^{(1)}) - \Psi_{r,\varepsilon,t}^{(2)}(x_{ij}^{(2)})\|_2. \tag{3.9}$$

See section 6 for examples.

Remark: See the appendix, sections 7.1 and 7.2, for a note on a technical issue concerning the computation of the eigenmaps.

3.3 Symmetrizing the distance and the kernel

In the computation of the diffusion maps detailed in the previous section, 3.2, both the distance η_r and the diffusion kernel $P_{r,\varepsilon}^{(\alpha)}$ were not symmetric. These two issues can be remedied in the following ways. First, to address the issue concerning η_r, we can redefine the initial kernel given in equation (3.7) as follows:

$$\tilde{k}_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)}) = e^{-\left[\eta_r(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2 + \eta_r(x_{i'j'}^{(\alpha)}, x_{ij}^{(\alpha)})^2\right]/\varepsilon}$$

Secondly, we can symmetrize the normalized kernel $p_{r,\varepsilon}$ given in equation 3.8 by redefining it as:

$$\tilde{p}_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)}) = \frac{\tilde{k}_{r,\varepsilon}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})}{\tilde{\omega}_{r,\varepsilon}^{1/2}(x_{ij}^{(\alpha)})\tilde{\omega}_{r,\varepsilon}^{1/2}(x_{i'j'}^{(\alpha)})},$$

where $\tilde{\omega}_{r,\varepsilon}$ is defined the same as $\omega_{r,\varepsilon}$ except that it sums entries of $\tilde{k}_{r,\varepsilon}$.

4 Global coordinates

We can modify the local coordinates described in section 3 to search globally (geographically speaking) for spectral nearest neighbors, as opposed to the local search described earlier. Mathematically speaking, this is equivalent to taking $r = \infty$ above; however, we make a few algorithmic adjustments, as well as one mathematical adjustment due to memory considerations.

Since we are performing a global search, we can apply PCA once to the entire data set $\mathcal{X}^{(\alpha)}$; we denote this set as:

$$\mathcal{X}_{pca}^{(\alpha)} = \{\tilde{x}_{ij}^{(\alpha)} : (i,j) \in \mathcal{I}\} \subset \mathbb{R}^d.$$

As in the local version, we assume that this set has been normalized so that the variance in each dimension is one. We also take as many dimensions as are necessary to retain 99% of the energy.

For each pixel $x_{ij}^{(\alpha)}$ we now perform a global search for spectral neighbors. We take $r_{ij}^{(\alpha)} = \|\tilde{x}_{ij}^{(\alpha)}\|_2$ to be the radius of the spectral ball that any neighbor must fall within, and we also let k_0 denote the maximum number of neighbors we are willing to allow (this is mainly due to memory considerations on the computer). Formally, we define:

$$\mathcal{I}_{r_{ij}^{(\alpha)}}(\tilde{x}_{ij}^{(\alpha)}) = \{(i',j') \in \mathcal{I} : \|\tilde{x}_{i'j'}^{(\alpha)} - \tilde{x}_{ij}^{(\alpha)}\|_2 \leq r_{ij}^{(\alpha)}\}$$

and

$\mathcal{I}_{k_0}(\tilde{x}_{ij}^{(\alpha)}) = \{(i',j') \in \mathcal{I} : \tilde{x}_{i'j'}^{(\alpha)}$ is one of the k_0 closest spectral neighbors of $\tilde{x}_{ij}^{(\alpha)}\}$.

The neighbors of $x_{ij}^{(\alpha)}$ are then given by:

$$\mathcal{N}_{r_{ij}^{(\alpha)},k_0}(x_{ij}^{(\alpha)}) = \{x_{i'j'}^{(\alpha)} \in \mathcal{X}^{(\alpha)} : (i',j') \in \mathcal{I}_{r_{ij}^{(\alpha)}}(\tilde{x}_{ij}^{(\alpha)}) \cap \mathcal{I}_{k_0}(\tilde{x}_{ij}^{(\alpha)})\}.$$

As in section 3, we now compute the average vector of $\mathcal{N}_{r_{ij}^{(\alpha)},k_0}(x_{ij}^{(\alpha)})$, as well as its covariance matrix, $C_{ij}^{(\alpha)}$. Let $\bar{x}_{ij}^{(\alpha)}$ denote the average vector, and let $\Lambda_{\infty,k_0}(x_{ij}^{(\alpha)})$ and $\mathcal{V}_{\infty,k_0}(x_{ij}^{(\alpha)})$ denote the eigenvalues and eigenvectors of $C_{ij}^{(\alpha)}$, respectively. We then have the following map:

$$x_{ij}^{(\alpha)} \mapsto \left(\bar{x}_{ij}^{(\alpha)}, \Lambda_{\infty,k_0}(x_{ij}^{(\alpha)}), \mathcal{V}_{\infty,k_0}(x_{ij}^{(\alpha)})\right). \tag{4.1}$$

We also have the maps η_{∞,k_0} and $d_{\text{total } \eta_{\infty,k_0}}$, which are identical to η_r and $d_{\text{total } \eta_r}$, respectively, except for the fact that the new maps use the global coordinates defined in (4.1). See section 6 for examples.

4.1 Diffusion maps

As in section 3.2, we can compute the diffusion maps based on η_{∞,k_0}. The details are exactly same, except that η_r is replaced with η_{∞,k_0}. We denote the resulting diffusion similarity measure by $\rho_{\infty,k_0,\varepsilon,t}$.

5 Comparing nearest neighbors

To detect change one can compare the nearest neighbors of $x_{ij}^{(1)}$ to the nearest neighbors of $x_{ij}^{(2)}$. Let $\mathcal{N}_k(x_{ij}^{(\alpha)})$ denote the k closest spectral neighbors to $x_{ij}^{(\alpha)}$, as measured by the ℓ^2 distance, and let $\mathcal{I}_k(x_{ij}^{(\alpha)})$ be their corresponding coordinates. We first determine the common neighbors of $x_{ij}^{(1)}$ and $x_{ij}^{(2)}$; that is, we compute the set:

$$\mathcal{I}_k(x_{ij}^{(1)}, x_{ij}^{(2)}) = \mathcal{I}_k(x_{ij}^{(1)}) \cap \mathcal{I}_k(x_{ij}^{(2)}).$$

The remaining neighbors are then paired so as to minimize the total spectral difference. More specifically, define

$$k'_{ij} = k - |\mathcal{I}_k(x_{ij}^{(1)}, x_{ij}^{(2)})|,$$

and let

$$\widetilde{\mathcal{N}}_k(x_{ij}^{(\alpha)}) = \{x_{i'j'}^{(\alpha)} \in \mathcal{N}_k(x_{ij}^{(\alpha)}) : (i',j') \notin \mathcal{I}_k(x_{ij}^{(1)}, x_{ij}^{(2)})\} = \{x_l^{(ij;\,\alpha)} : l = 1, \ldots, k'_{ij}\}.$$

We also set $S_{k'}$ to be the set of permutations on k' elements. Finally, for each coordinate pair $(i,j) \in \mathcal{I}$, we compute

$$d_{nn}(x_{ij}^{(1)}, x_{ij}^{(2)}) = \min_{\sigma \in S_{k'_{ij}}} \sum_{l=1}^{k'_{ij}} \|x_l^{(ij;\,1)} - x_{\sigma(l)}^{(ij;\,2)}\|_2 \tag{5.1}$$

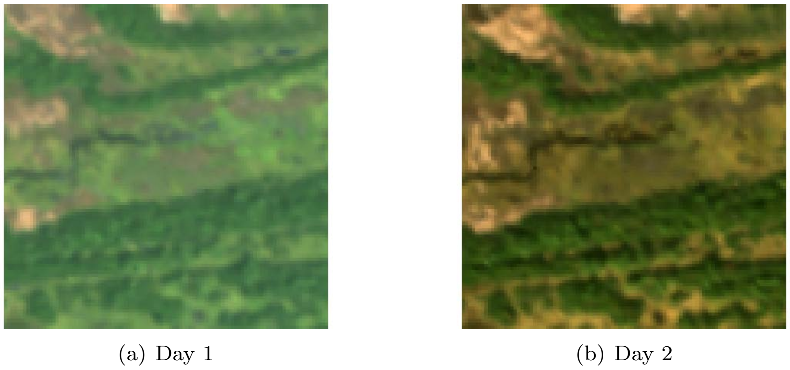
The nearest neighbor distance map, d_{nn}, gives a measure of how much change occurred between $x_{ij}^{(1)}$ and $x_{ij}^{(2)}$.

6 Examples

6.1 Patch A

We consider the following 100 × 100 × 113 patch, for which pseudo-color images from day one and day two are depicted in figure 1.

Figure 1: Pseudo-color images of patch A



(a) Day 1

(b) Day 2

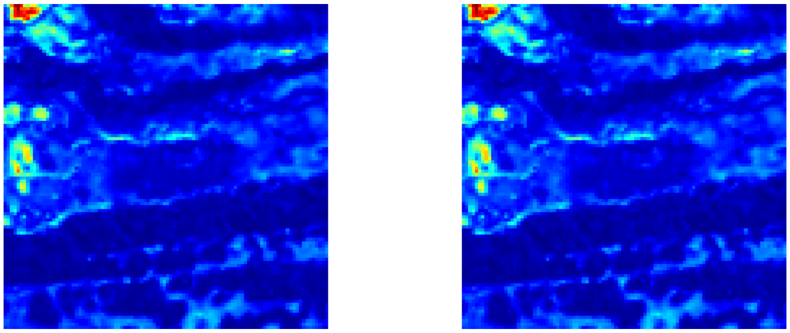
6.1.1 Simple Solutions

The inner product map, d_{ip}, defined in (2.1), and the ℓ^2 distance map, d_{ℓ^2}, defined in (2.2), are shown in figure 2.

The total ℓ^2 distance map, $d_{\text{total } \ell^2}$, defined in (2.3), is given in figure 3.

The PCA ℓ^2 distance map, $d_{\text{PCA } \ell^2}$, defined in (2.4), was computed with $d = 7$ dimensions, the minimum number required so that both days' mappings retain at least 99% of the information. The result is given in figure 4.

Figure 2: d_{ip} and d_{ℓ^2} similarity measures for patch A



(a) d_{ip}

(b) d_{ℓ^2}

Figure 3: $d_{\text{total }\ell^2}$ similarity measure for patch A

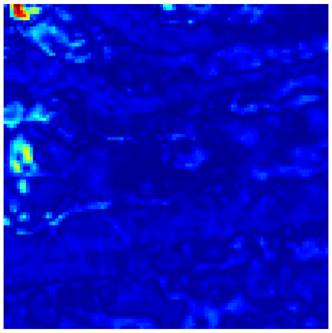
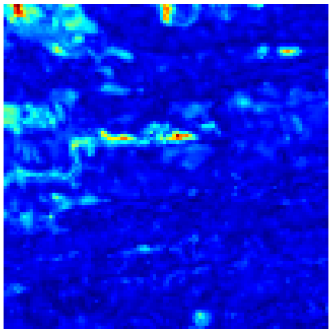


Figure 4: $d_{\text{PCA }\ell^2}$ similarity measure for patch A



6.1.2 Local coordinates for change detection

We ran the algorithm described in section 3 with $r = 2$. The $d_{\text{total }\eta_r}$ graph similarity measure, defined in (3.6), is shown in figure 5.

For the $\rho_{r,\varepsilon,t}$ diffusion similarity measure, defined in (3.9), we computed the diffusion maps using $\varepsilon = 1$, $t = 1$, and retained $s = 20$ eigenmaps; the results are given in figure 6. Note that the sign of the eigenvectors was not corrected (as described in section 7.1) for this particular experiment.

We also computed the $\rho_{r,\varepsilon,t}$ diffusion similarity measure using

$$\varepsilon = \varepsilon^{(\alpha)} = \text{median}\left\{\eta_r(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2 : (i,j),(i',j') \in \mathcal{I}\right\},$$

$t = 1$, and retained $s = 6$ eigenmaps; the results are given in figure 7.

Figure 5: $d_{\text{total }\eta_r}$ ($r = 2$) local similarity measure for patch A

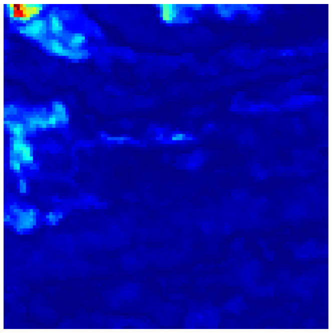


Figure 6: $\rho_{r,\varepsilon,t}$ ($r = 2$, $\varepsilon = 1$, $t = 1$, $s = 20$) local diffusion similarity measure for patch A

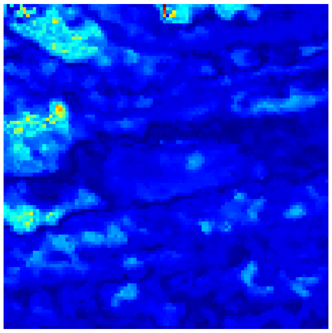
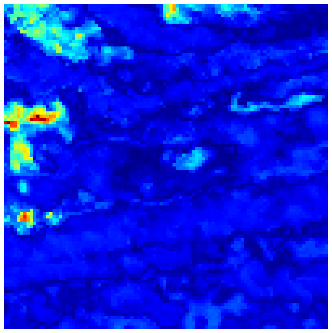


Figure 7: $\rho_{r,\varepsilon,t}$ ($r = 2$, $\varepsilon = \varepsilon^{(\alpha)}$, $t = 1$, $s = 6$) local diffusion similarity measure for patch A



6.1.3 Global coordinates

We ran the algorithm described in section 4 with $k_0 = 50$. The total difference map, $d_{\text{total }\eta_{\infty,k_0}}$ for patch A is given in figure 8.

We also computed the corresponding diffusion maps, with

$$\varepsilon = \varepsilon^{(\alpha)} = \text{median}\left\{\eta_{\infty,k_0}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2 : (i,j), (i',j') \in \mathcal{I}\right\},$$

varying values of t, and $s = 6$. We synced the eigenmaps according to the method outlined in section 7.1. The resulting diffusion similarity maps, $\rho_{\infty,k_0,\varepsilon,t}$ are given in figure 9.

Thirdly, we computed the corresponding diffusion maps by symmetrizing the kernel according to section 3.3. We used $\tilde{\varepsilon}^{(\alpha)}$ defined correspondingly as:

$$\varepsilon = \tilde{\varepsilon}^{(\alpha)} = \text{median}\left\{\eta_{\infty,k_0}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2 + \eta_{\infty,k_0}(x_{i'j'}^{(\alpha)}, x_{ij}^{(\alpha)})^2 : (i,j), (i',j') \in \mathcal{I}\right\},$$

and varied the values of t. We used $s = 100$ eigenmaps, which were synchronized according the method outlined in section 7.2. The resulting diffusion similarity maps, $\rho_{\infty,k_0,\varepsilon,t}$ are given in figure 10.

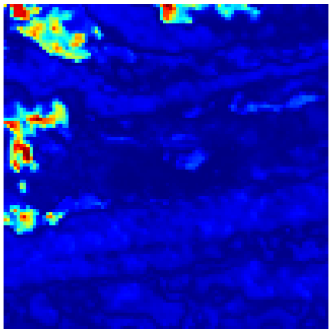
6.1.4 Comparing nearest neighbors

We ran the algorithm described in section 5 with $k = 20$ and computed the map d_{nn} for patch A. The results are given in figure 11.

6.2 Patch B

We consider the following 96 × 119 × 113 patch, for which pseudo-color images from day one and day two are depicted in figure 12.

Figure 8: $d_{\text{total }\eta_{\infty,k_0}}$ ($k_0 = 50$) global similarity map for patch A



6.2.1 Simple Solutions

The inner product map, d_{ip}, defined in (2.1), and the ℓ^2 distance map, d_{ℓ^2}, defined in (2.2), are shown in figure 13.

The total ℓ^2 distance map, $d_{\text{total }\ell^2}$, defined in (2.3), is given in figure 14.

The PCA ℓ^2 distance map, $d_{\text{PCA }\ell^2}$, defined in (2.4), was computed with $d = 6$ dimensions, the minimum number required so that both days' mappings retain at least 99% of the information. The result is given in figure 15.

6.2.2 Local coordinates for change detection

We ran the algorithm described in section 3 with $r = 2$. The $d_{\text{total }\eta_r}$ graph similarity measure, defined in (3.6), is shown in figure 16.

For the $\rho_{r,\varepsilon,t}$ diffusion similarity measure, defined in (3.9), we computed the diffusion maps using $\varepsilon = 1$, $t = 1$, and retained $s = 45$ eigenmaps; the results are given in figure 17. Note that the sign of the eigenvectors was not corrected (as described in section 7.1) for this particular experiment.

We also computed the $\rho_{r,\varepsilon,t}$ diffusion similarity measure using

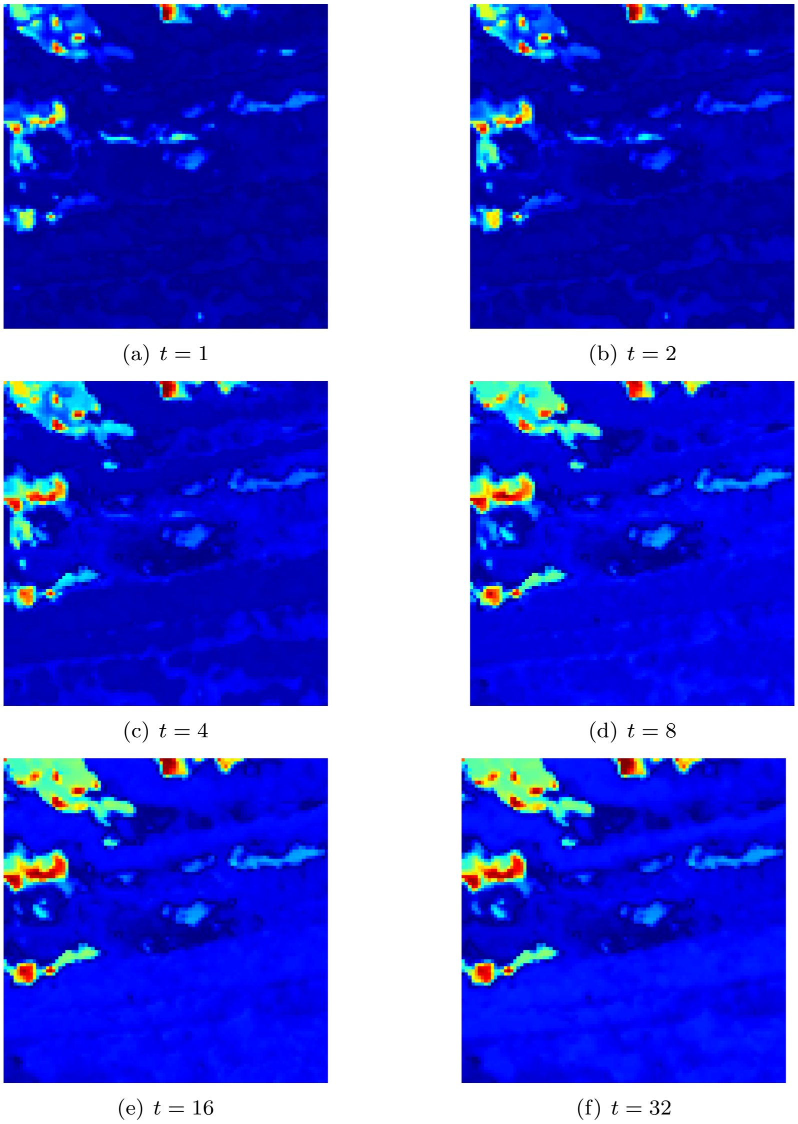
$$\varepsilon = \varepsilon^{(\alpha)} = \text{median}\left\{\eta_r(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2 : (i,j),(i',j') \in \mathcal{I}\right\},$$

$t = 1$, and retained $s = 4$ eigenmaps; the results are given in figure 18.

6.2.3 Global coordinates

We ran the algorithm described in section 4 with $k_0 = 50$. The total difference map, $d_{\text{total }\eta_{\infty,k_0}}$ for patch B is given in figure 19.

Figure 9: $\rho_{\infty,k_0,\varepsilon,t}$ ($k_0 = 50$, $\varepsilon = \varepsilon^{(\alpha)}$, t varying, $s = 6$) global diffusion similarity maps for patch A



(a) $t = 1$

(b) $t = 2$

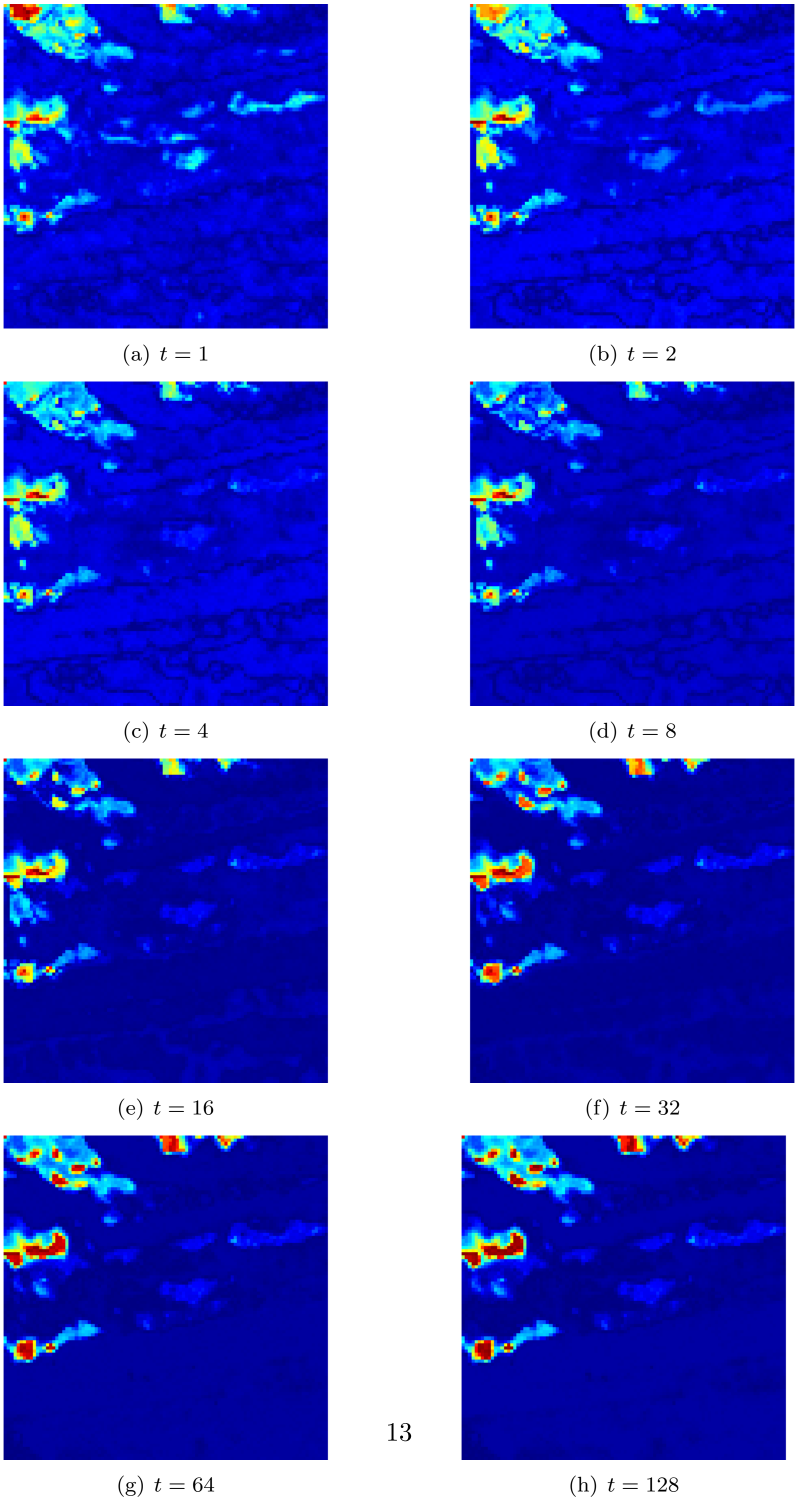
(c) $t = 4$

(d) $t = 8$

(e) $t = 16$

(f) $t = 32$

Figure 10: Symmetrized and synchronized $\rho_{\infty,k_0,\varepsilon,t}$ ($k_0 = 50$, $\varepsilon = \tilde{\varepsilon}^{(\alpha)}$, t varying, $s = 100$) global diffusion similarity maps for patch A



(a) $t = 1$

(b) $t = 2$

(c) $t = 4$

(d) $t = 8$

(e) $t = 16$

(f) $t = 32$

(g) $t = 64$

(h) $t = 128$

Figure 11: d_{nn} nearest neighbor similarity measure for patch A

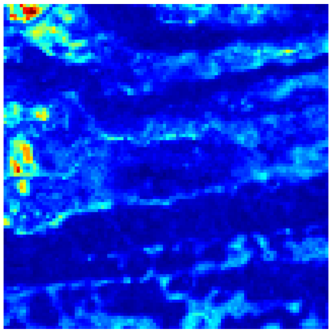
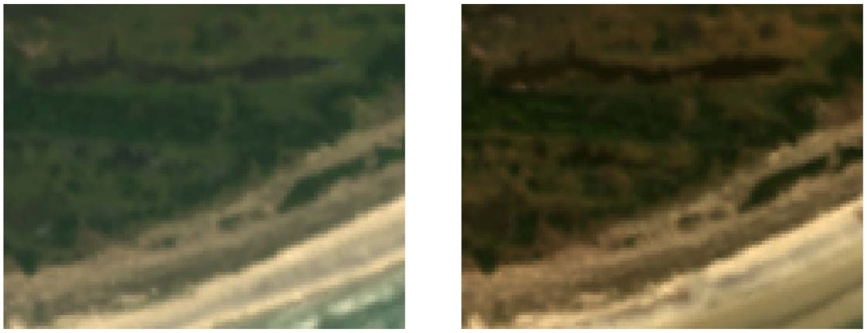


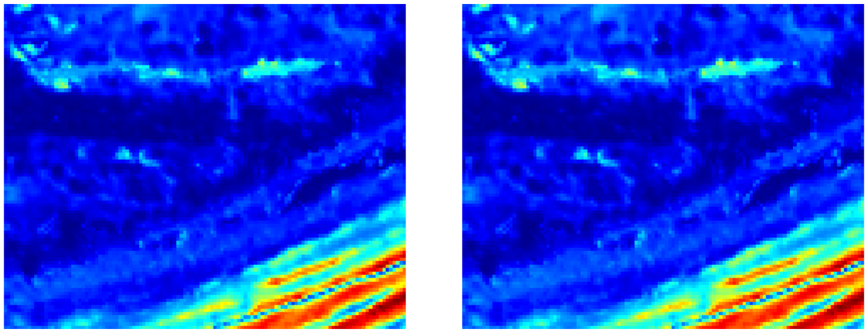
Figure 12: Pseudo-color images of patch B



(a) Day 1

(b) Day 2

Figure 13: d_{ip} and d_{ℓ^2} similarity measures for patch B



(a) d_{ip}

(b) d_{ℓ^2}

Figure 14: $d_{\text{total } \ell^2}$ similarity measure for patch B

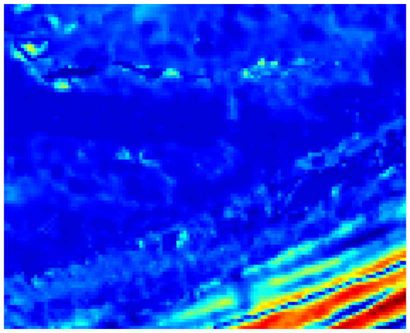


Figure 15: $d_{\text{PCA } \ell^2}$ similarity measure for patch B

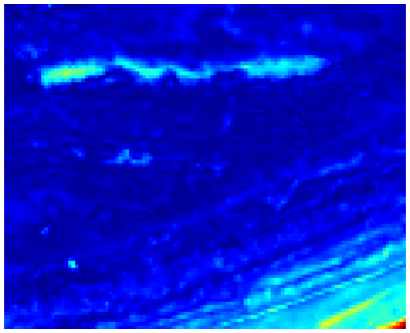


Figure 16: $d_{\text{total } \eta_r}$ $(r = 2)$ graph similarity measure for patch B

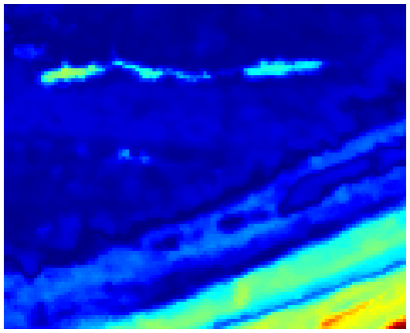


Figure 17: $\rho_{r,\varepsilon,t}$ ($r = 2$, $\varepsilon = 1$, $t = 1$, $s = 45$) local diffusion similarity measure for patch B

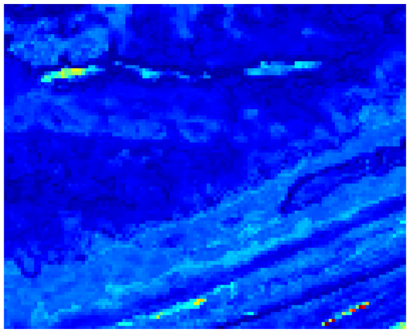
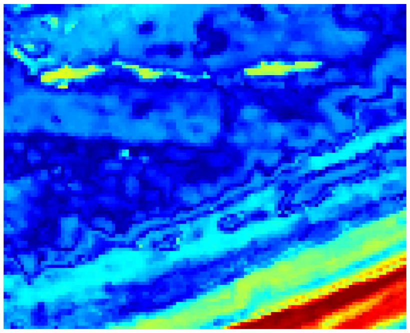


Figure 18: $\rho_{r,\varepsilon,t}$ ($r = 2$, $\varepsilon = \varepsilon^{(\alpha)}$, $t = 1$, $s = 4$) local diffusion similarity measure for patch A

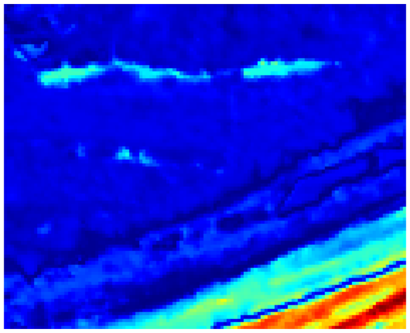


We also computed the corresponding diffusion maps, with

$$\varepsilon = \varepsilon^{(\alpha)} = \text{median}\left\{\eta_{\infty,k_0}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2 : (i,j),(i',j') \in \mathcal{I}\right\},$$

varying values of t, and $s = 8$. We synced the eigenmaps according to the method outlined in section 7.1. The resulting diffusion similarity maps, $\rho_{\infty,k_0,\varepsilon,t}$ are given in figure 20.

Figure 19: $d_{\text{total }\eta_{\infty,k_0}}$ ($k_0 = 50$) global similarity map for patch B



6.2.4 Comparing nearest neighbors

We ran the algorithm described in section 5 with $k = 20$ and computed the map d_{nn} for patch B. The results are given in figure 21.

6.3 Alternate patch B

We modified patch B, day 1, by placing a water pixel at row 37, column 58. This pixel was originally vegetation. We then reran some of the experiments. Results are given below.

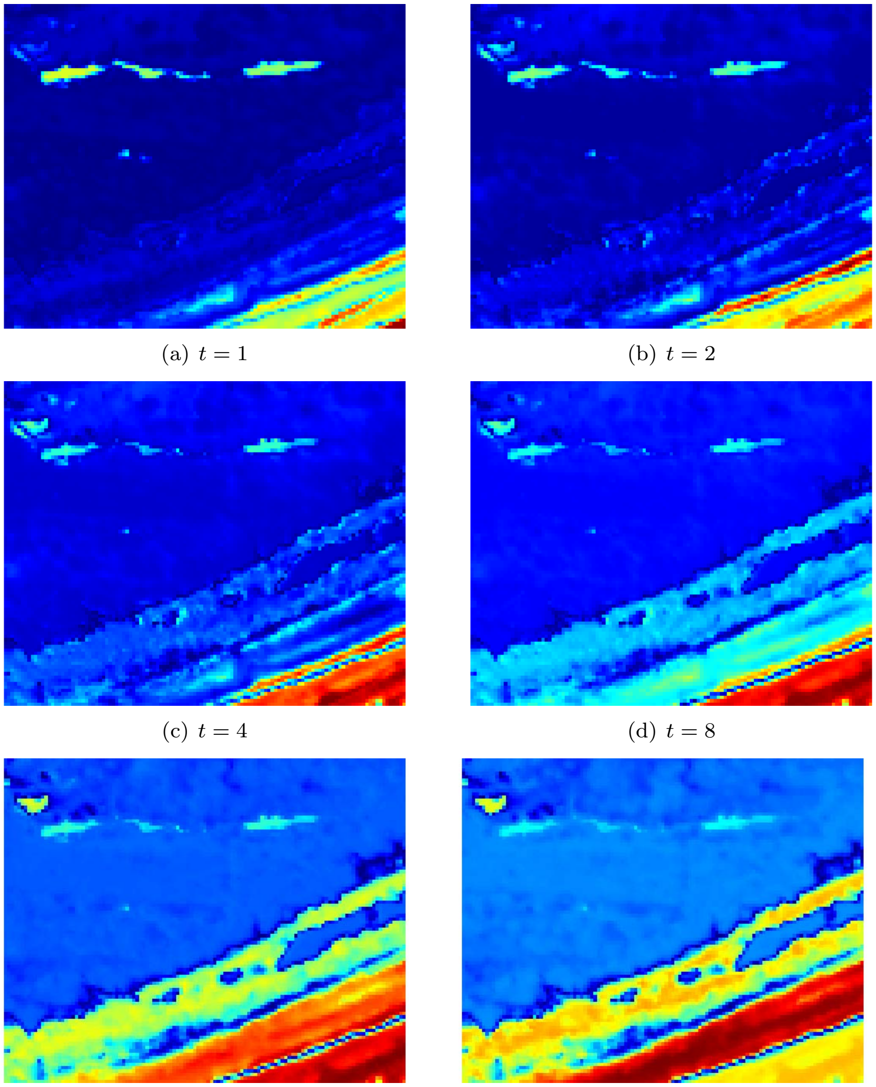
6.3.1 Simple solutions

The PCA ℓ^2 distance map, $d_{\text{PCA }\ell^2}$, defined in (2.4), was computed with $d = 6$ dimensions, the minimum number required so that both days' mappings retain at least 99% of the information. The result is given in figure 22.

6.3.2 Global coordinates

We used the same settings as described in section 6.2.3. The resulting $d_{\text{total }\eta_{\infty,k_0}}$ map is given in figure 23, while the diffusion similarity maps $\rho_{\infty,k_0,\varepsilon,t}$ are given in figure 24.

Figure 20: $\rho_{\infty,k_0,\varepsilon,t}$ ($k_0 = 50$, $\varepsilon = \varepsilon^{(\alpha)}$, t varying, $s = 8$) global diffusion similarity maps for patch B



(a) $t = 1$

(b) $t = 2$

(c) $t = 4$

(d) $t = 8$

(e) $t = 16$

(f) $t = 32$

Figure 21: d_{nn} nearest neighbor similarity measure for patch B

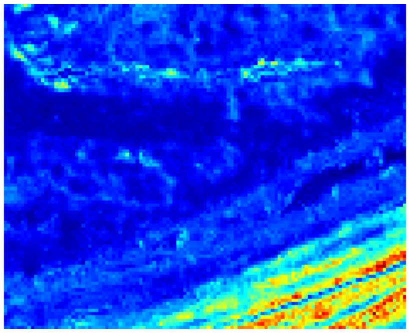
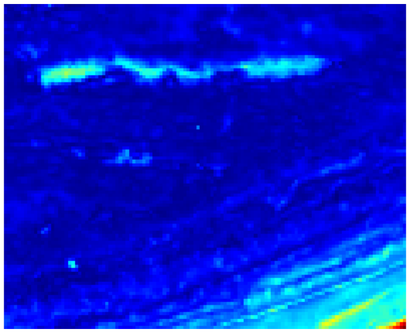


Figure 22: $d_{\mathrm{PCA}\ \ell^2}$ similarity measure for alternate day 1 patch B



We also computed the diffusion maps by symmetrizing the kernel according to section 3.3. We used $\tilde{\varepsilon}^{(\alpha)}$ defined correspondingly as:

$$\varepsilon = \tilde{\varepsilon}^{(\alpha)} = \text{median}\left\{\eta_{\infty,k_0}(x_{ij}^{(\alpha)}, x_{i'j'}^{(\alpha)})^2 + \eta_{\infty,k_0}(x_{i'j'}^{(\alpha)}, x_{ij}^{(\alpha)})^2 : (i,j), (i',j') \in \mathcal{I}\right\},$$

and varied the values of t. We used $s = 100$ eigenmaps, which were synchronized according the method outlined in section 7.2. The resulting diffusion similarity maps, $\rho_{\infty,k_0,\varepsilon,t}$ are given in figure 25.

Figure 23: $d_{\text{total }\eta_{\infty,k_0}}$ ($k_0 = 50$) global similarity map for alternate day 1 patch B

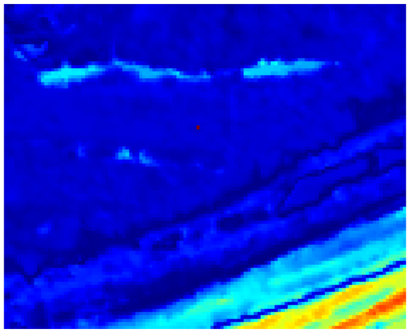
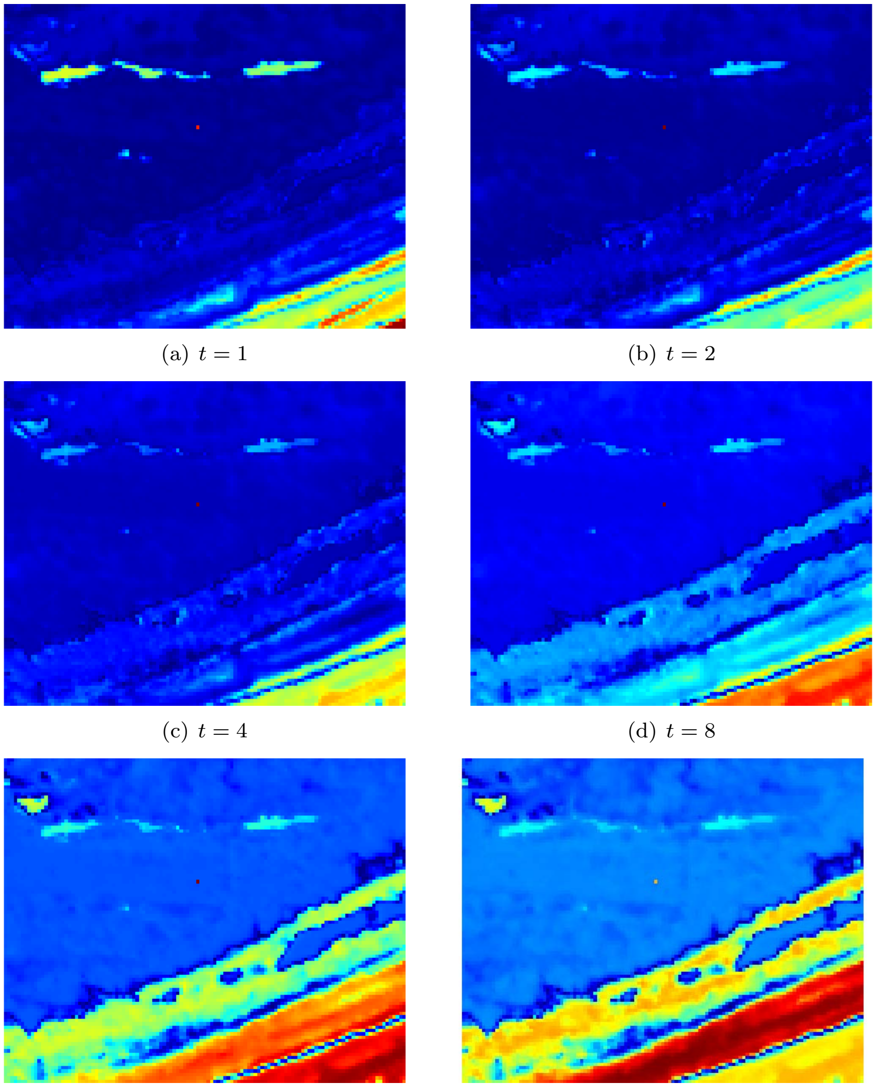


Figure 24: $\rho_{\infty,k_0,\varepsilon,t}$ ($k_0 = 50$, $\varepsilon = \varepsilon^{(\alpha)}$, t varying, $s = 8$) global diffusion similarity maps for alternate day 1 patch B



(a) $t = 1$

(b) $t = 2$

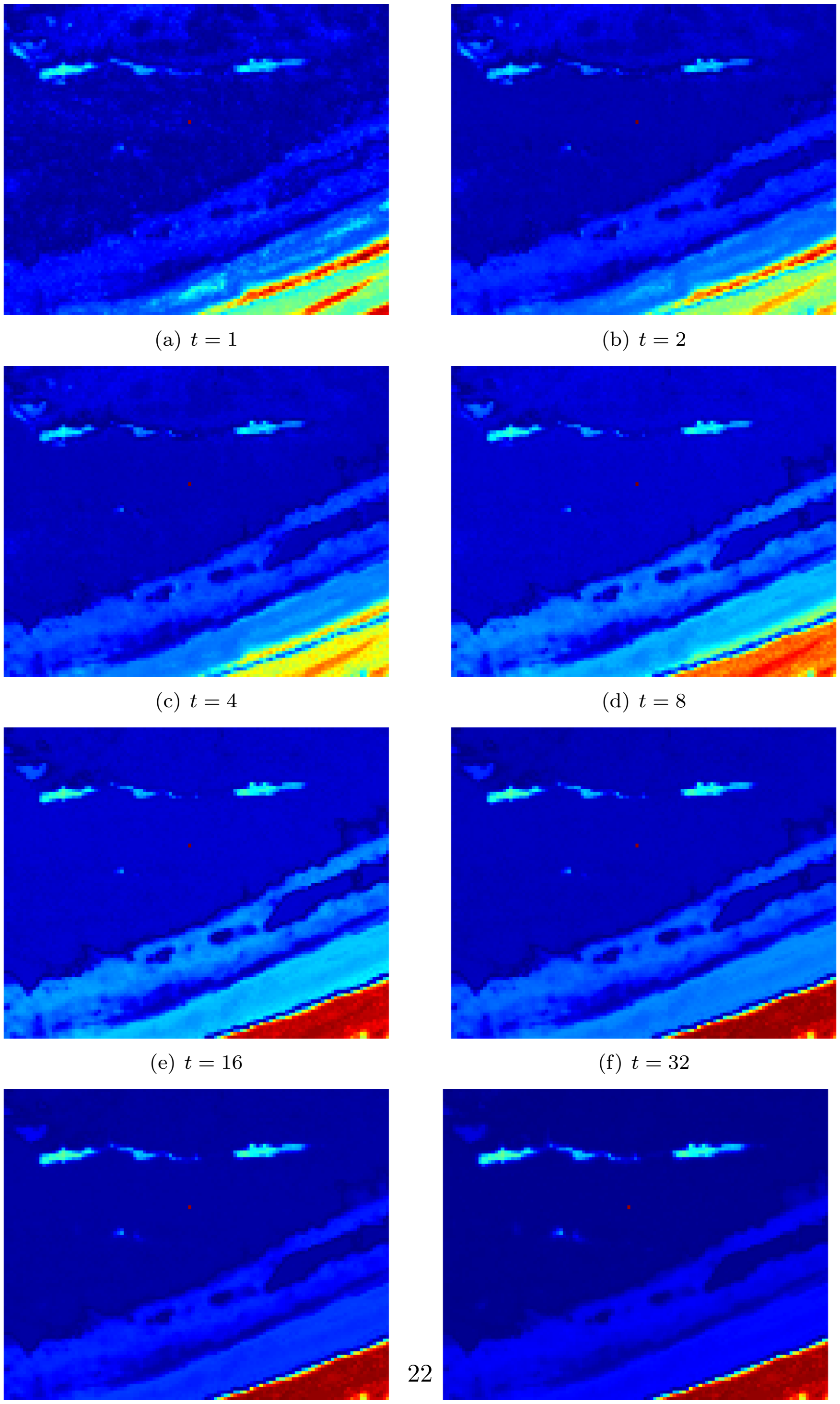
(c) $t = 4$

(d) $t = 8$

(e) $t = 16$

(f) $t = 32$

Figure 25: Symmetrized and synchronized $\rho_{\infty,k_0,\varepsilon,t}$ ($k_0 = 50$, $\varepsilon = \tilde{\varepsilon}^{(\alpha)}$, t varying, $s = 100$) global diffusion similarity maps for alternate day 1 patch B



(a) $t = 1$

(b) $t = 2$

(c) $t = 4$

(d) $t = 8$

(e) $t = 16$

(f) $t = 32$

(g) $t = 64$

(h) $t = 128$

7 Appendix

7.1 Synchronizing the sign of eigenmaps

When MATLAB computes the eigenmaps in the diffusion maps algorithm, they are unique up to a sign change. When dealing with one data set, this is not much of an issue. However, when dealing with two data sets, and subsequently examining the ℓ^2-difference between each days' diffusion coordinates, this can lead to inaccuracies when trying to detect change. In order to address this issue, we have implemented the following method for correction.

Given the two sets of eigenvectors, we compute the following s inner products:

$$\vartheta(i) = \frac{\langle \psi_i^{(1)}, \psi_i^{(2)} \rangle}{\|\psi_i^{(1)}\|_2 \|\psi_i^{(2)}\|_2}, \qquad i = 1, \ldots, s.$$

Note that the range of ϑ is $[-1, 1]$, and that if $|\vartheta(i)|$ is near one, then the eigenvectors $\psi_i^{(1)}$ and $\psi_i^{(2)}$ are highly correlated. However, if $\vartheta(i) < 0$ as well, then when computing the ℓ^2 difference, these highly correlated eigenvectors will actually appear far apart. Therefore, we make the following adjustment to the signs of the eigenvectors of day 2:

$$|\vartheta(i)| > \theta \text{ and } \vartheta(i) < 0 \quad \Longrightarrow \quad \psi_i^{(2)} \mapsto -\psi_i^{(2)},$$

where θ is some preset measure of correlation.

Note: In the examples that employ this method, we have set $\theta = 0.75$.

7.2 Synchronizing the eigenmaps for maximum correlation

We can take the method outlined above one step further, and synchronize the eigenmaps in order to maximize their correlation. To do this, we compute the following permutation (note that the eigenvectors $\psi_i^{(\alpha)}$ have norm one):

$$\sigma = \arg\max_{\sigma' \in S_s} \sum_{i=1}^{s} |\langle \psi_i^{(1)}, \psi_{\sigma'(i)}^{(2)} \rangle|$$

We then map the day two eigenvectors as follows:

$$\psi_i^{(2)} \mapsto \psi_{\sigma(i)}^{(2)}.$$

Finally, after permuting the day two eigenvectors, we make the appropriate sign changes if the angle between two eigenvectors is greater than 90 degrees:

$$\langle \psi_i^{(1)}, \psi_i^{(2)} \rangle < 0 \quad \Longrightarrow \quad \psi_i^{(2)} \mapsto -\psi_i^{(2)}.$$

We also permute the eigenvalues accordingly. First:

$$\xi_i^{(2)} \mapsto \xi_{\sigma(i)}^{(2)}.$$

And then secondly, after the permutation:

$$\xi_i^{(1)} \xi_i^{(2)} < 0 \quad \implies \quad \xi_i^{(2)} \mapsto -\xi_i^{(2)}.$$

Appendix D: *Hyperspectral Explorer* User Manual

USER MANUAL

HYPERSPECTRAL EXPLORER

Version 1.0

CONFIDENTIAL AND PROPRIETARY

PLAIN SIGHT
SYSTEMS

Table of Contents

1 How this manual is organized

The goal of this manual is to guide a first-time user through the installation, setup and first steps in using HyperSpectral Explorer, gaining progressively knowledge of the capabilities of the software, and ultimately to master all its capabilities.

2 Scope & Objectives

The main objectives of HyperSpectral Explorer are to provide the following:

2.i Data normalization, compression and denoising

Interface for initial manipulation of the collected data.
The type of manipulations we are referring to may include data denoising, compression, normalization, computation of svd of the data, and possibly other data selection/enanchement algorithms.

2.ii Data Exploration

Interface for algorithms for data exploration
This involves the development of a user-friendly, but still powerful (for the most knowledgeable user) interface to various algorithms (either developed by the group or described in papers etc...) for the exploration of data: among these, e.g., projections on particular coordinates (svd, random projections), mainly for dimensionality reduction purposes; nonlinear transformations (kernel eigenmap) for nonlinear dimension reduction/parametrization; data (labelled/unlabelled) dynamic exploration in various sets of coordinates (linear/non-linear)

2.iii Learning/discovering structures in the data

Interface for algorithms for discrimination
Development of a user-friendly interface to algorithms, developed by the group or elsewhere, supervised or unsupervised, for data clustering and other discrimination tasks. Learning algorithms may include Local Discriminant Bases for supervised feature selection, fast hierarchical clustering techniques, unsupervised or partially supervised, non-linear CART based on Laplacian eigenfunctions etc...The output of these algorithms will be avilable to the user for further manipulations and exploration.

2.iv Data sonification

This part of the user interface will allow the user to map the original data, or any of its representations in feauture spaces, to various sound spaces, using tunable maps and a choice among different sound spaces. This allows the exploration of the data through sounds, allowing the sonification of features as well as, possibly, the discovery of features through sound exploration.
This capabilities may not be available on all installations or on all configurations.

3 Installing the HyperSpectral Explorer

3.i Requirements

HyperSpectral Explorer requires Windows 2000 or XP operating system.

Version 1.0 of HyperSpectral Explorer also requires a licenced version of Matlab 6 (rev 13) to be installed locally, with license server either local or remote. Use of remote Matlab servers has not been tested.

The current version of Hyperspectral Explorer is not compatible with versions of Matlab superior to version 6 (release 13), since Mathworks does not guarantee forward compatibility of the Matlab library functions.

The user needs administrator privileges to install and run HyperSpectral Explorer. Moreover the user needs the maximum COM/DCOM/COM+ privileges: contact your system administrator if you are not sure whether you have these and/or how to set them.

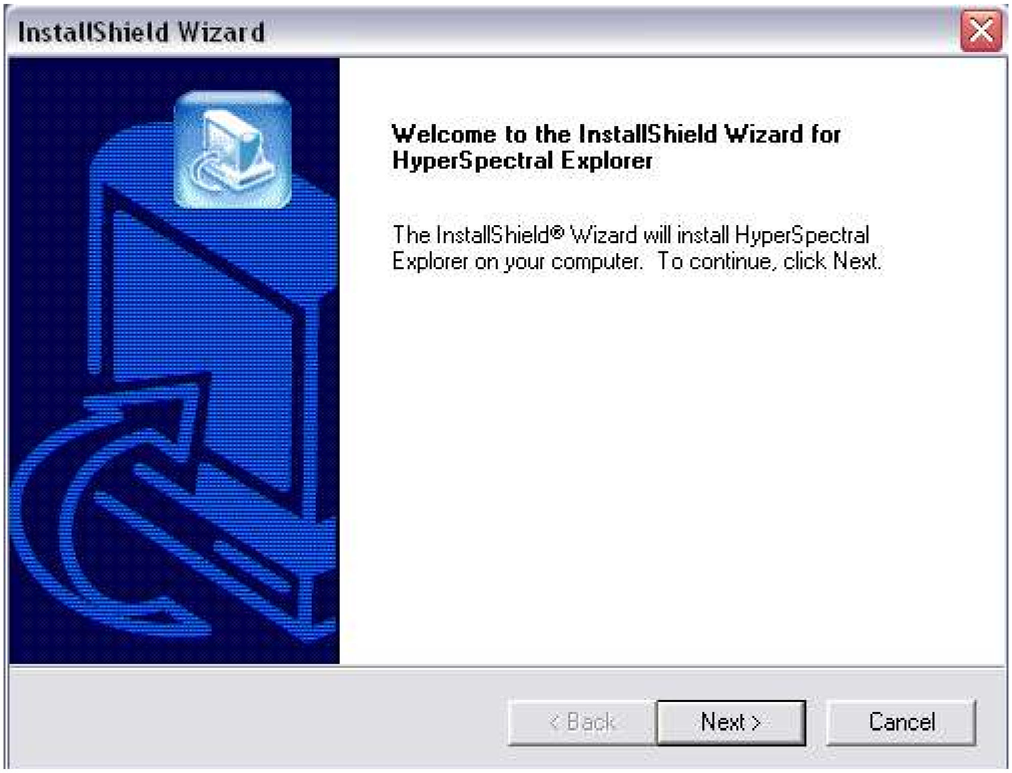
HyperSpectral Explorer also requires Microsoft XML Parser to be installed. This is already installed during most installations of Windows XP, otherwise it needs to be installed prior to the installation of HyperSpectral Explorer. The installation package can be found at:

http://www.microsoft.com/downloads/details.aspx?FamilyID=c0f86022-2d4c-4162-8fb8-66bfc12f32b0&displaylang=en

or by Googling “Microsoft MSXML parser download”.

3.ii Starting the installation process

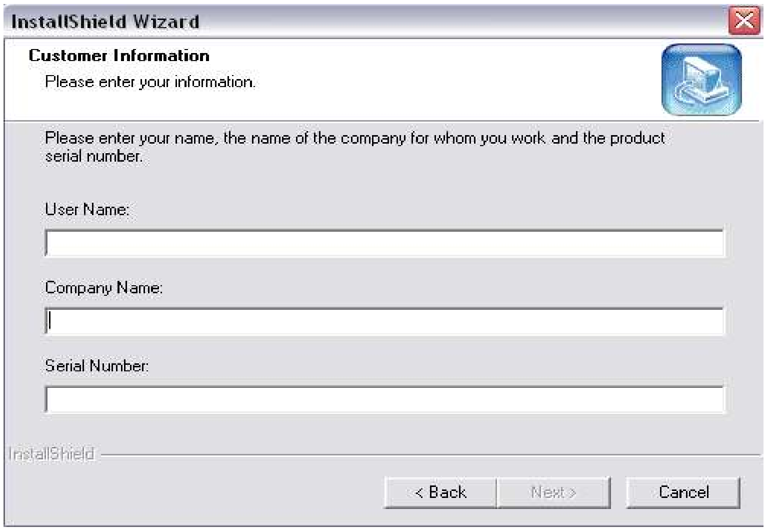
On the installation disk, simply run the executable file called setup.exe. This will start the installation process.



Welcome to the InstallShield Wizard for HyperSpectral Explorer

The InstallShield® Wizard will install HyperSpectral Explorer on your computer. To continue, click Next.

3.iii Guide through the installation process



Customer Information

Please enter your information.

Please enter your name, the name of the company for whom you work and the product serial number.

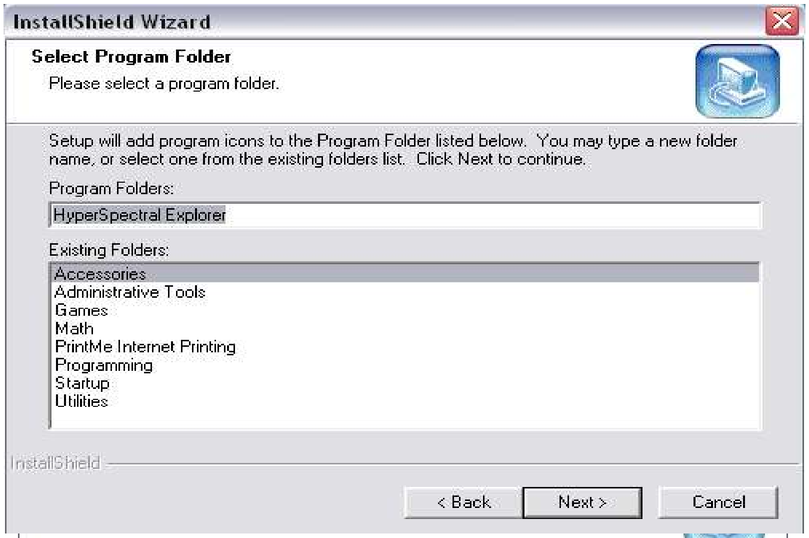
User Name:

Company Name:

Serial Number:

Enter you name, company name, and any serial number.

Select a directory where to install



Select Program Folder

Please select a program folder.

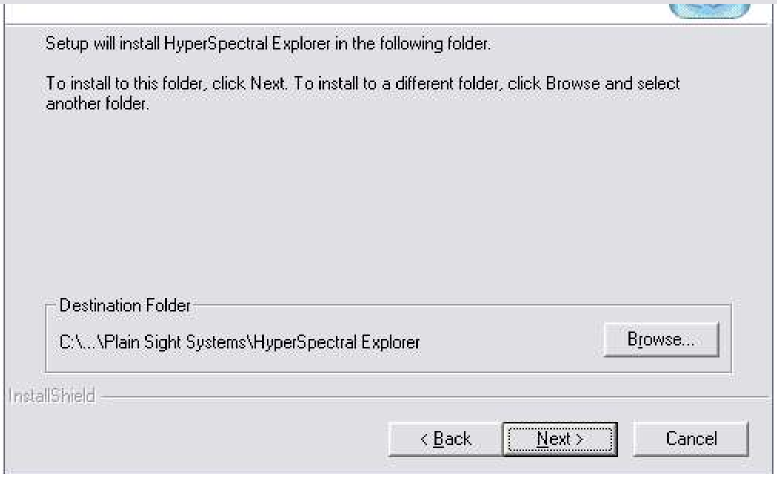
Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing folders list. Click Next to continue.

Program Folders:

HyperSpectral Explorer

Existing Folders:

- Accessories
- Administrative Tools
- Games
- Math
- PrintMe Internet Printing
- Programming
- Startup
- Utilities



Setup will install HyperSpectral Explorer in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\...\Plain Sight Systems\HyperSpectral Explorer

Select a folder where to install the shortcut to Hyperspectral Explorer.

The setup program will then install the Hyperspectral Explorer files in the specified directory, add the shortcuts to the specified folder and add a shortcut on the desktop.

The software is now ready to run.

4 Starting HyperSpectral Explorer

4.i Starting the installation process

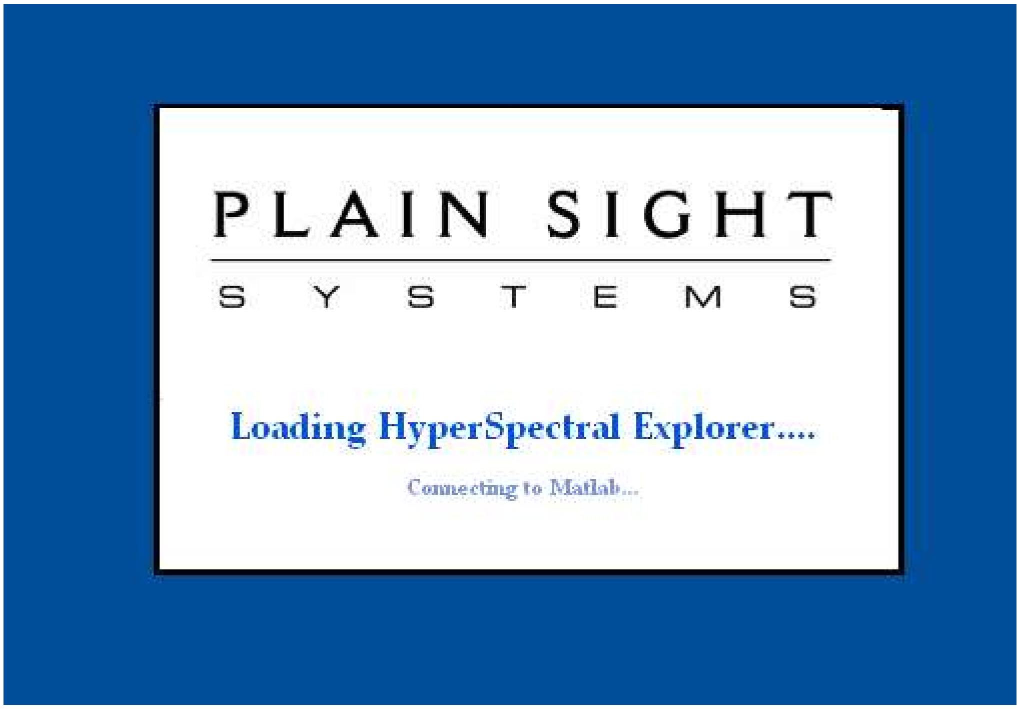
Simply click on the icon in the PlainSight System folder created in the Start Menu during the installation, or on the icon placed on the desktop:

HyperSpectral
Explorer

If neither icon is available, the executable will be in the installation directory and is named prHyperSpectralExplorer.exe

4.ii Loading

During the loading phase, the HyperSpectral Explorer will show progress messages on splash screen (colors may very depending on the system settings):



PLAIN SIGHT
SYSTEMS

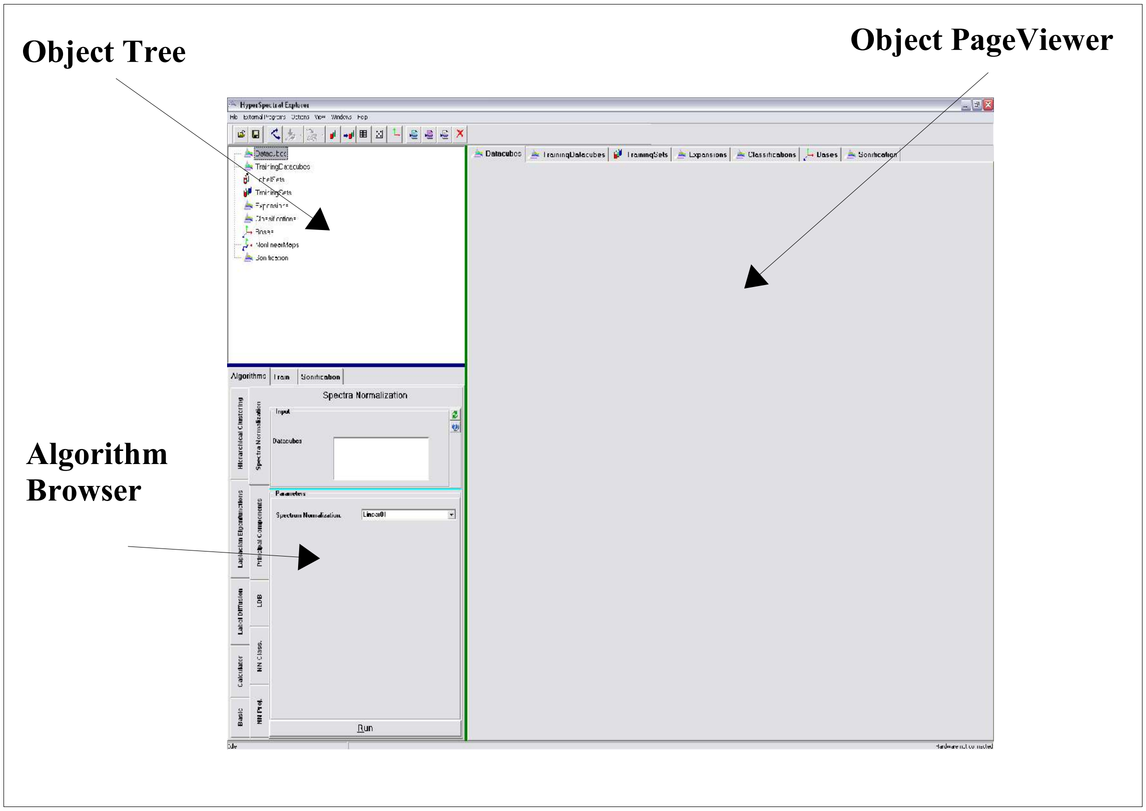
Loading HyperSpectral Explorer....

Connecting to Matlab...

Loading time will vary according to the speed of the machine, memory available, security settings and Matlab installation type (on the machine or remote). During loading, HyperSpectral Explorer will connect to the currently registered version of Matlab (either on that machine or a remote machine). This is usually the longest procedure during loading.

5 Organization of the User Interface

5.i General Overview



Spectrum Normalization: Linear01

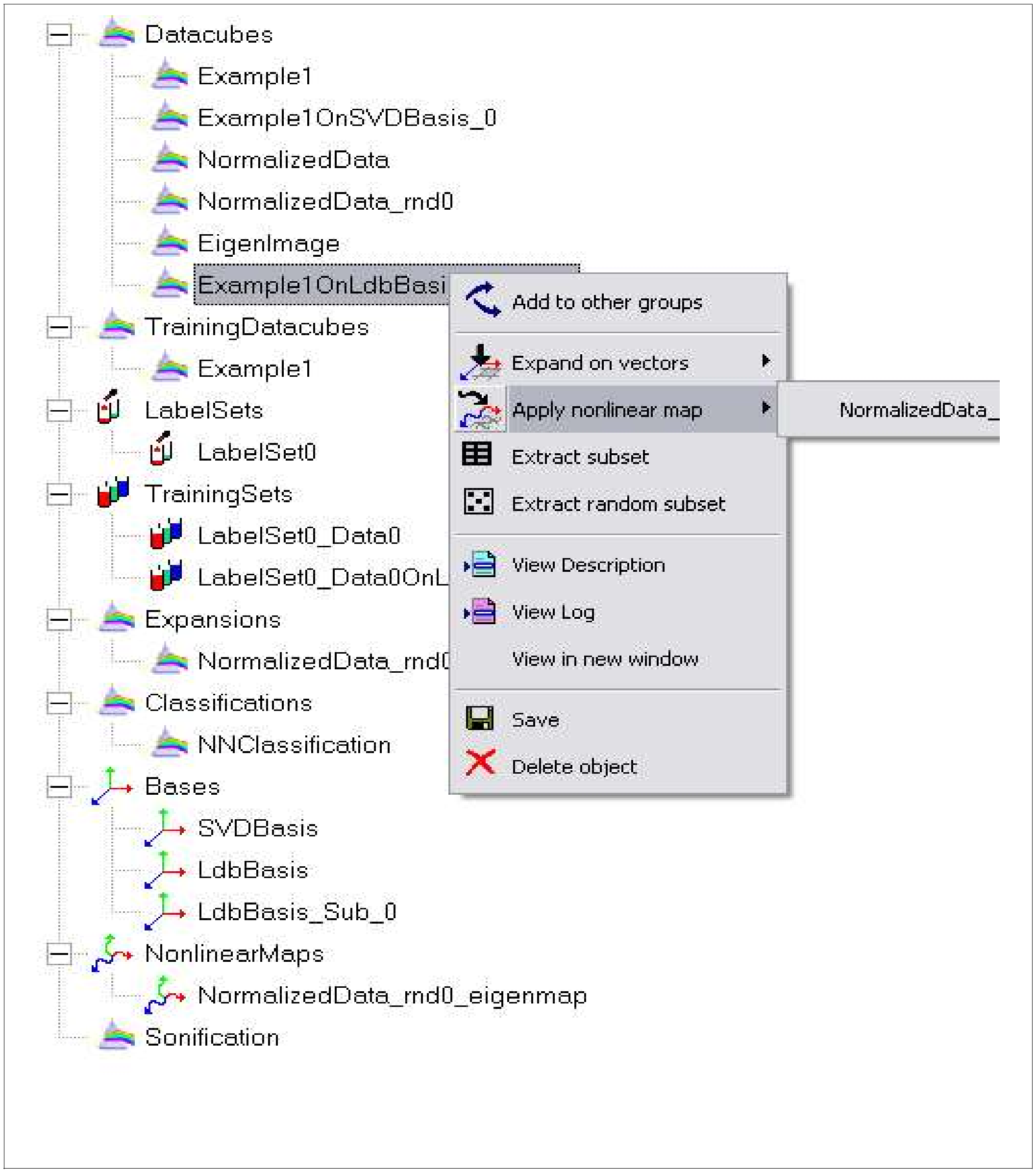
Below is how HyperSpectral Explorer presents itself after loading (minor differences may depend on your system configuration. the screen is divided into three main parts:

The three parts can be resized by using the vertical green slidebar separating the Object PageViewer from the Object Tree and the Algorithm Browser, while the horizontal blue slidebar can be used to redistribute the space between the Object Tree and the Algorithm Browser.

At the top of the window there is a menu with some standard functionalities, while below the menu there is a very useful toolbar, whose buttons allow to perform most actions on the objects in the workspace.

5.ii The Object Tree

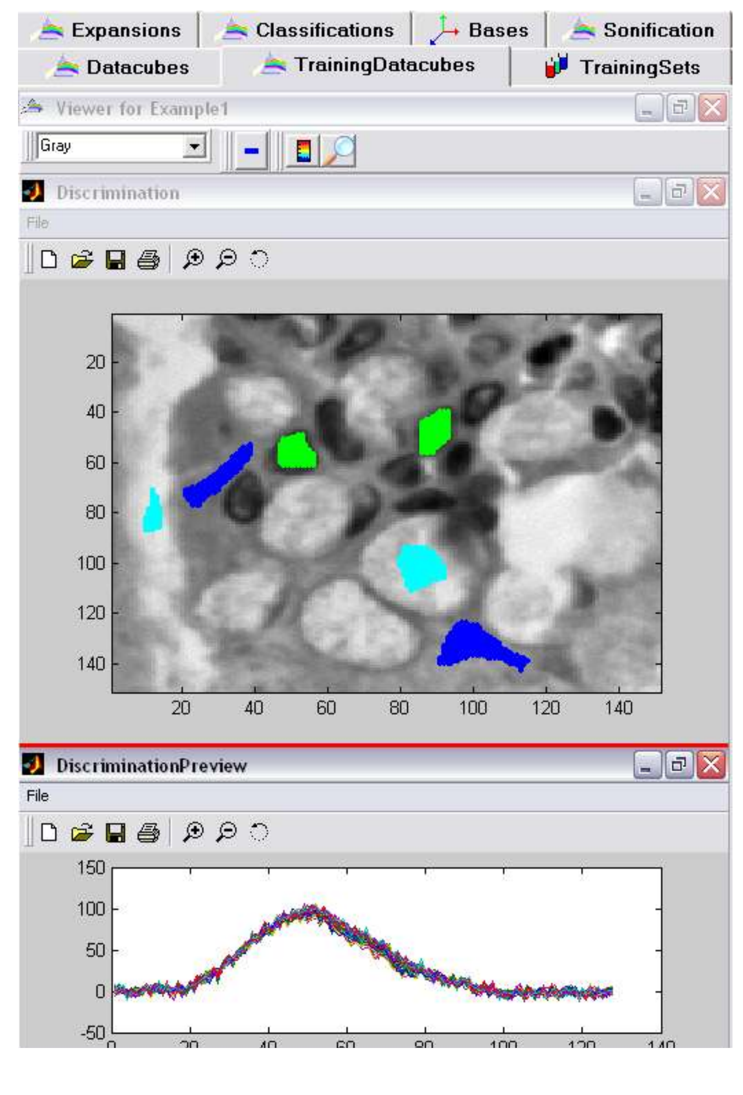
The Object Tree, top left, represents the objects currently available to the user, divided into conceptual categories or groups. Initially the tree is empty, only the object catogories appear. When objects are loaded, manipulated and created, they appear in the branches of tree. Right-clicking on various objects in the tree will activate pop-up menus that allow the user to perform actions on the objects and make them interact.



The object tree at the end of the tutorial in this manual may look as in the picture below. Many objects have been created during the analysis of one initial data cube “Example1”. Some of then represent data cubes, but the third dimension has a meaning which is not a spectral frequency, but a probability, or a coefficient of a projection onto some features. There are also orthogonal bases among the objects created, as well as nonlinear maps, and training sets. Popup menus are accessed by selecting an element and right-clicking.

5.iii The Object PageViewer

The Object PageViewer is a set of tabs, divided conceptually in the same categories as the objects in the object tree, on each of which certain types of objects are visualized. The categories are a (possibly strict, depending on system configuration) subset of:



- *Preview*: when connected to the hardware, on this page the user will be able to preview the images captured by the instrument, collect pictures, set the parameters for the collection, etc...This option may or not be available depending on the connected hardware and on the installation options for the hardware driver.
- *Datacubes*: on this page datacubes and other data are represented. For example when the user loads a datacube, a viewer for the loaded object is displayed on this page. Essentially on this page there should be all the “working objects” the user wants to manipulate.
- *Training*: on this page the datasets that the user chooses to use for training are displayed. These datasets are usually imported from the set of “working objects” in the *datacubes* page. Objects on the *datacubes* page can be moved onto the *training* page by right-clicking on the *datacube* object in the **object tree** and choosing the action “Add to training set”.
- *Classifications*: the data on this page is the results of a classification task. It can be a set of vectors with corresponding labels, represented with points in a vector space, with colors indicating the corresponding labels.
- *Bases*: the data on this page represents basis objects, e.g. the basis vectors resulting

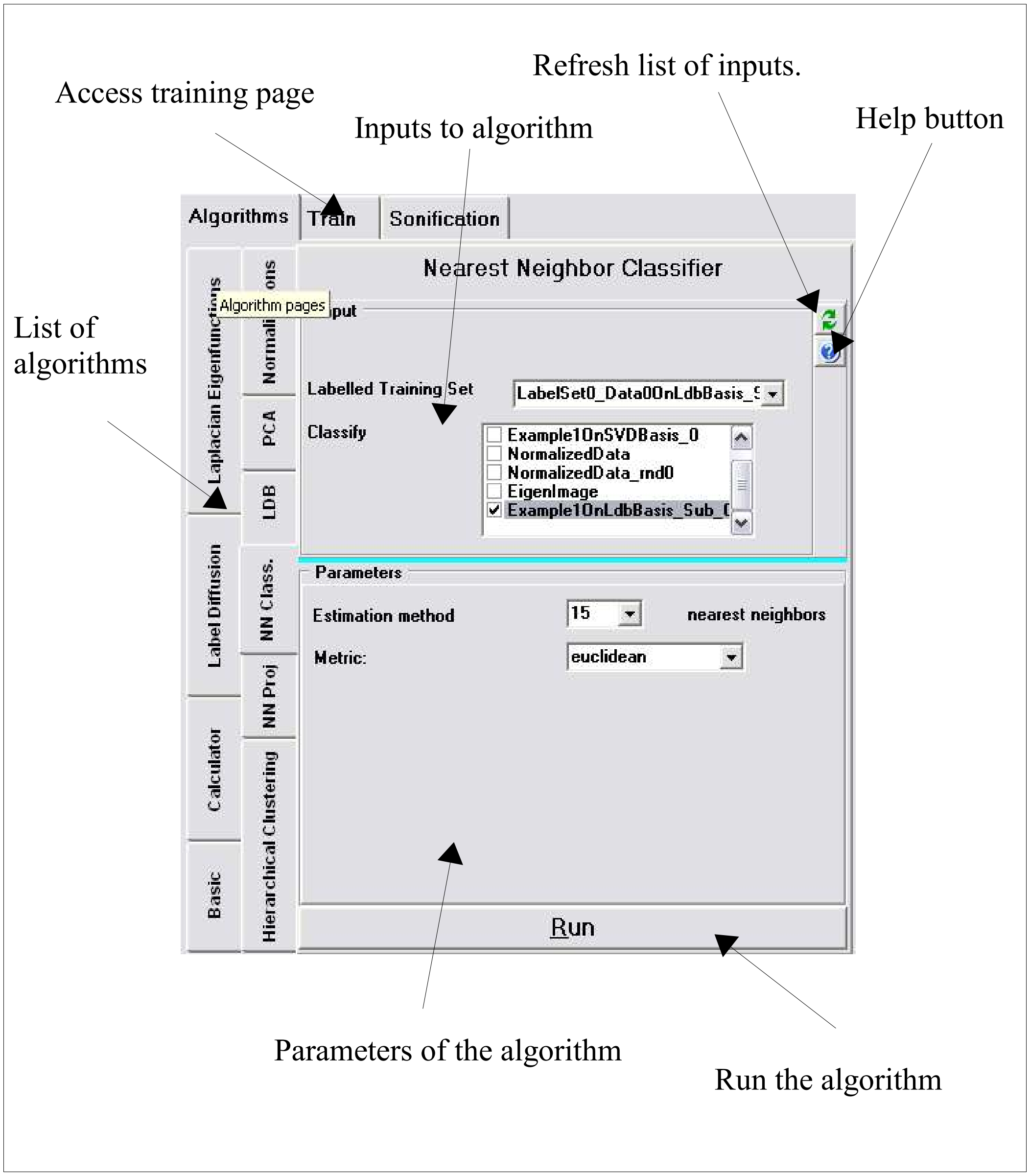
from the run of a linear discriminant basis algorithm.

- *Expansions*: the data on this page represents the image of some *datacube* object under a linear or nonlinear mapping. For example the projection of a *datacube* object onto its SVD basis.
- *Sonification*: the data on this page is for sonification purposes. Not available in all configurations.

5.iv The Algorithm Browser

The Algorithm Browser, bottom left, is a multi-page collection of user interfaces linking to various algorithms the user can apply on the available objects. It contains various pages, depending on system (hardware/software) configuration, among the following:

- *Preview*: parameters and actions related to the preview and collection of new datacubes. Not available in all configurations.
- *Train*: interface for building and merging labelled training sets.
- *Algorithms*: interface to various algorithms that act on *datacube* objects and/or on *training* objects.
- *Sonification*: interface to various sonification algorithms.



Nearest Neighbor Classifier

Input

Labelled Training Set: LabelSet0_Data00nLdbBasis_S

Classify

- ☐ Example1OnSVDBasis_0
- ☐ NormalizedData
- ☐ NormalizedData_rnd0
- ☐ EigenImage
- ☑ Example1OnLdbBasis_Sub_(

Parameters

Estimation method: 15 nearest neighbors

Metric: euclidean

6 Main Menu

6.i File Menu

Open

Open an existing Matlab, HSE, or PND file into HyperSpectral Explorer.

Save

Saves all the files in the current workspace to HSE format files.

Save Matlab workspace

Saves all the variables currently loaded in Matlab to a Matlab file.

Exit

Exits from HyperSpectral Explorer.

6.ii External Programs Menu

Connect to Matlab

Connect to the Matlab server registered on the machine (local or network matlab server).

Disconnect from Matlab

Disconnect from the Matlab server HyperSpectral Explorer is currently connected to. This will cause all the variables currently in Matlab to be freed from memory and permantently lost.

Matlab – Reset random number generator

Resets the random number generator in Matlab. This affects the runs of randomized algorithms.

6.iii View Menu

View application log

Views the application log file, containing all the actions of the application since startup.

6.iv Windows Menu

Tile horizontally

Tiles the windows in the current page of the Object PageViewer horizontally.

Tile vertically

Tiles the windows in the current page of the Object PageViewer vertically.

6.v Help Menu

Help

Shows this help in a window in HyperSpectral Explorer.

About

Shows the program title, version, authors.

7 Getting Started: an example

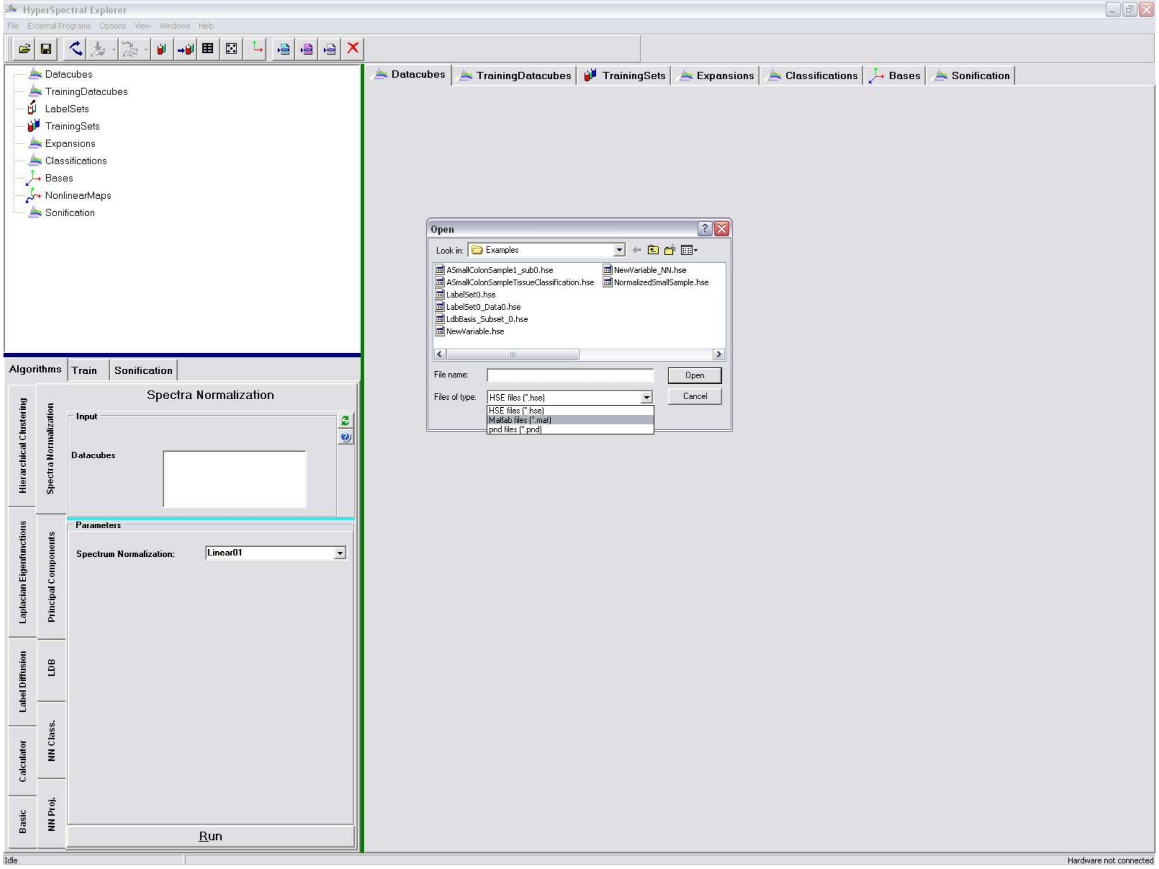
We show some capabilities of the HyperSpectral Explorer through an example.

7.i Starting up and loading a file

Start up the HyperSpectral Explorer

Click on the **Open** button at the top left (or select the Open command from the File menu)

The dialog box represented below will appear:



Spectrum Normalization: Linear01

Look in: Examples

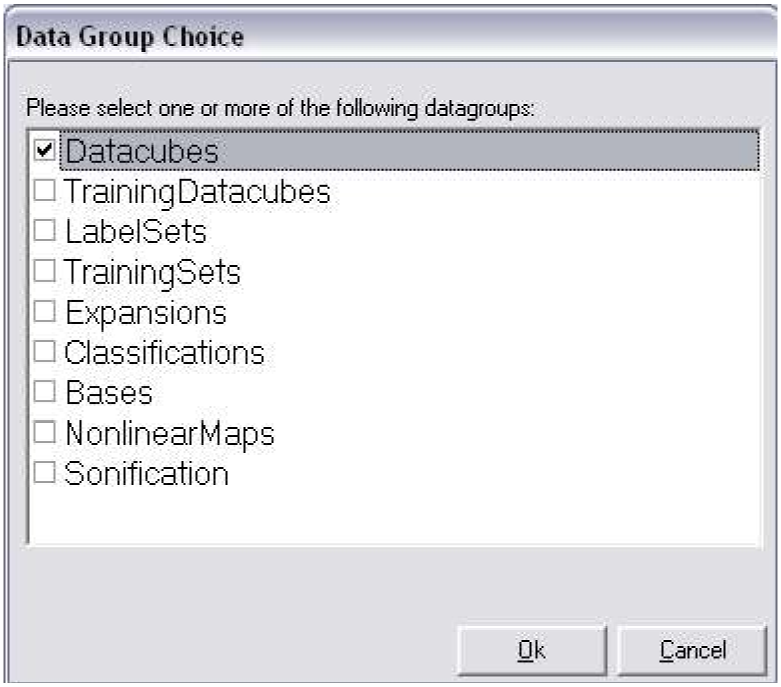
ASmallColonSample1_sub0.hse
ASmallColonSampleTissueClassification.hse
LabelSet0.hse
LabelSet0_Data0.hse
LdbBasis_Subset_0.hse
NewVariable.hse
NewVariable_NN.hse
NormalizedSmallSample.hse

Files of type: HSE files (*.hse)

HSE files (*.hse)
Matlab files (*.mat)
pnd files (*.pnd)

Go in the directory "Examples" in the installation directory of Hyperspectral Explorer.
Select "Matlab files" in the file type dropbox.
Choose the file named Example1.mat and click ok.

The dialog box below will appear, asking which group the loaded variable should be assigned to. Each object can be assigned to one or more data-groups, provided it is compatible with them. Data-groups are useful to divide the various loaded objects into classes with different properties, different roles in the algorithsm and different conceptual types.



Data Group Choice

Please select one or more of the following datagroups:

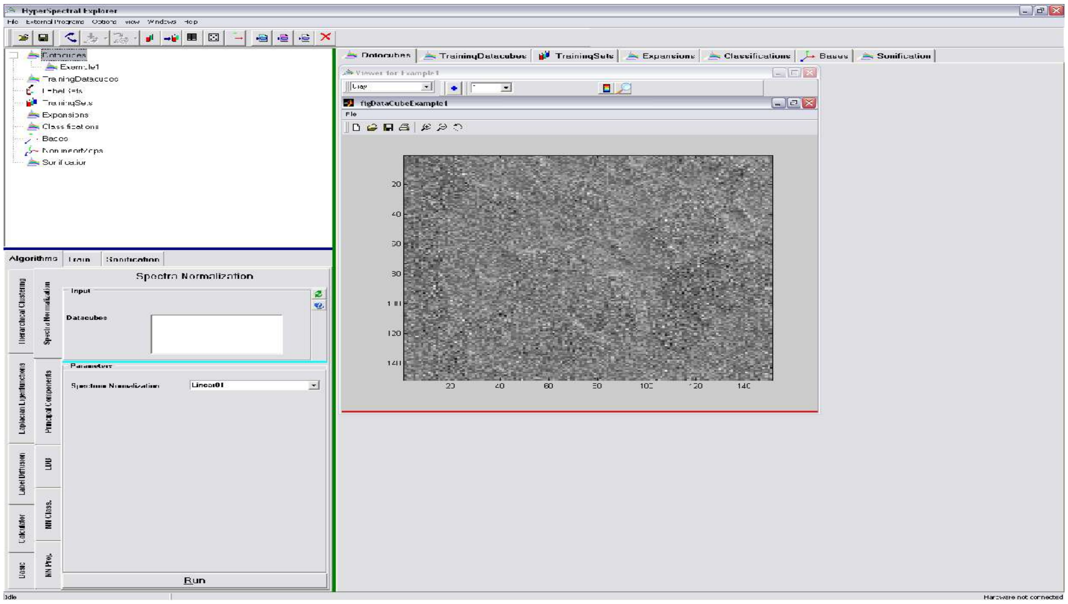
- ☑ Datacubes
- □ TrainingDatacubes
- □ LabelSets
- □ TrainingSets
- □ Expansions
- □ Classifications
- □ Bases
- □ NonlinearMaps
- □ Sonification

In the specific case, we assign the new variable to the “Datacubes” group. We will see later how to assign the variable to other groups.

Click Ok to load the data (this may take a few seconds depending on the specifications of your system, in particular cpu speed and available memory).

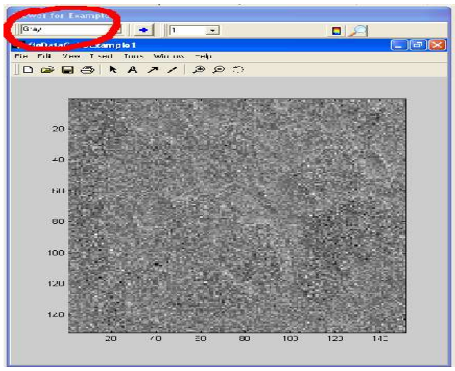
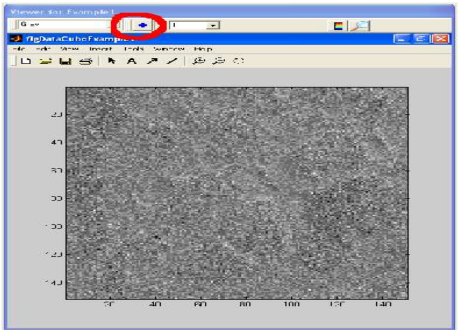
7.ii Browsing through the data

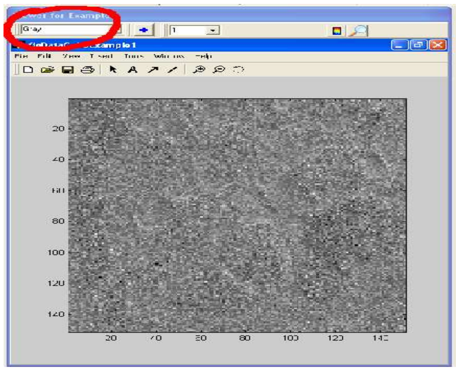
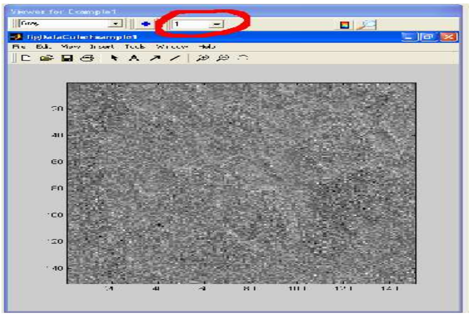
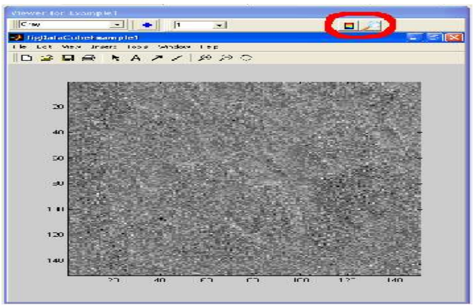
The file will be loaded, and object called Example1 will be added to the *Datacubes* branch in the object tree, and a Viewer for the object will be added to the *Datacubes* page in the Object PageViewer. The *algorithm browser* will switch to the *Datacubes* page:

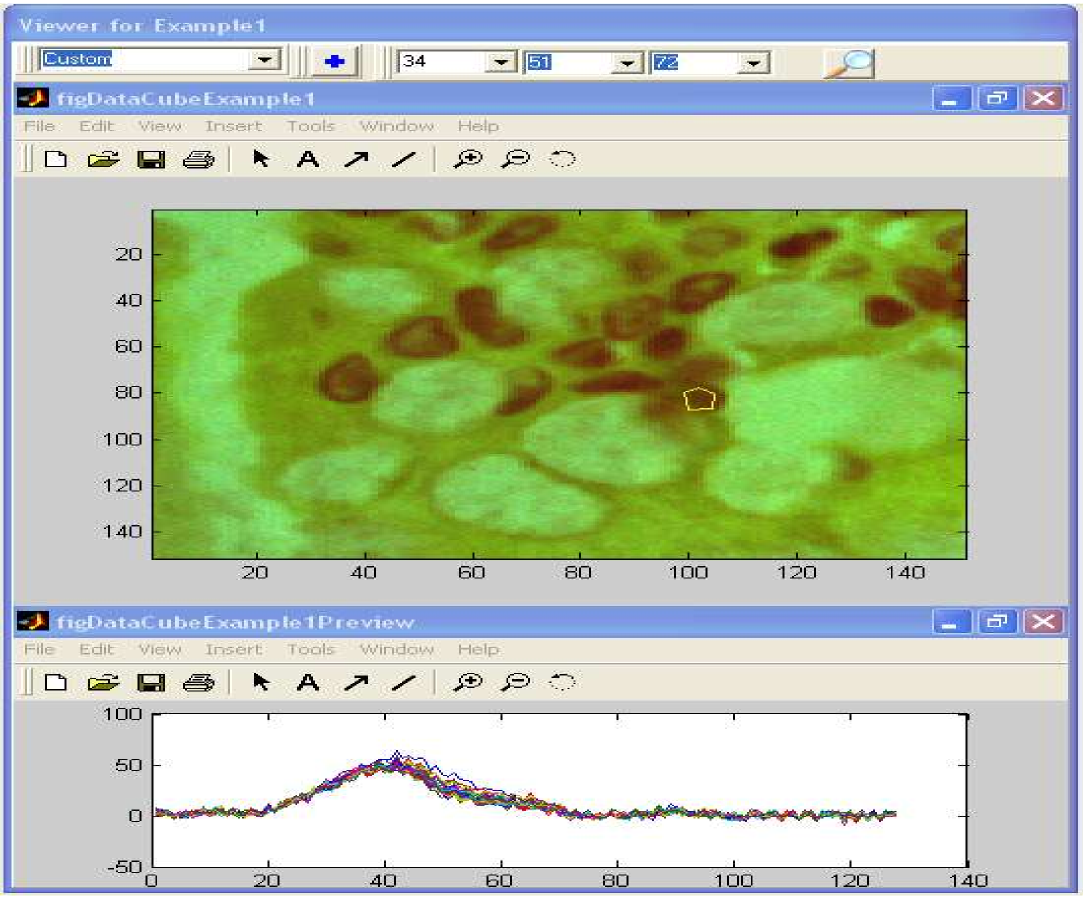


Spectrum Normalization Linear01

Let us browse through the data to see how the data cube browser works.

 Viewer for Example1 Gray figDataCubeExample1 File Edit View Insert Tools Window Help 20 40 60 80 100 120 140	The dropbox at the top left allows for different colouring of the datacube. A single slices can be viewed in gray scale, rgb, hsv and arbitrarily tuned color maps, while multiple spectral slices can be combined in two ways: three of them can be combined by mapping them to rgb (option 'custom'), or arbitrary linear combinations of slices can be taken with the 'equalizer' option.
 Viewer for Example1 Gray figDataCubeExample1 File Edit View Insert Tools Window Help 20 40 60 80 100 120 140	The highlighted button sums all the slices and displayes theirs sum in grayscale. For datacubes, this essentially shows the total white-light response.

	The dropbox at the top left allows for different colouring of the datacube. A single slices can be viewed in gray scale, rgb, hsv and arbitrarily tuned color maps, while multiple spectral slices can be combined in two ways: three of them can be combined by mapping them to rgb (option 'custom'), or arbitrary linear combinations of slices can be taken with the 'equalizer' option.
	This dropbox allows for visualizing different slices of the datacubes. One dropbox is available for 'gray', 'rgb', 'hsv','tuner' choices of coloring, while 3 are avaible when 'custom' is the choice of coloring
	The 'Show colorbar' button toggles the color scale on the side of picture on and off. The 'Zoom' button toggles on and off a window at the bottom of the viewer where the spectra of the selection is showed.



CAUTION: The red button for closing the pictures window inside the browser should never be used to close

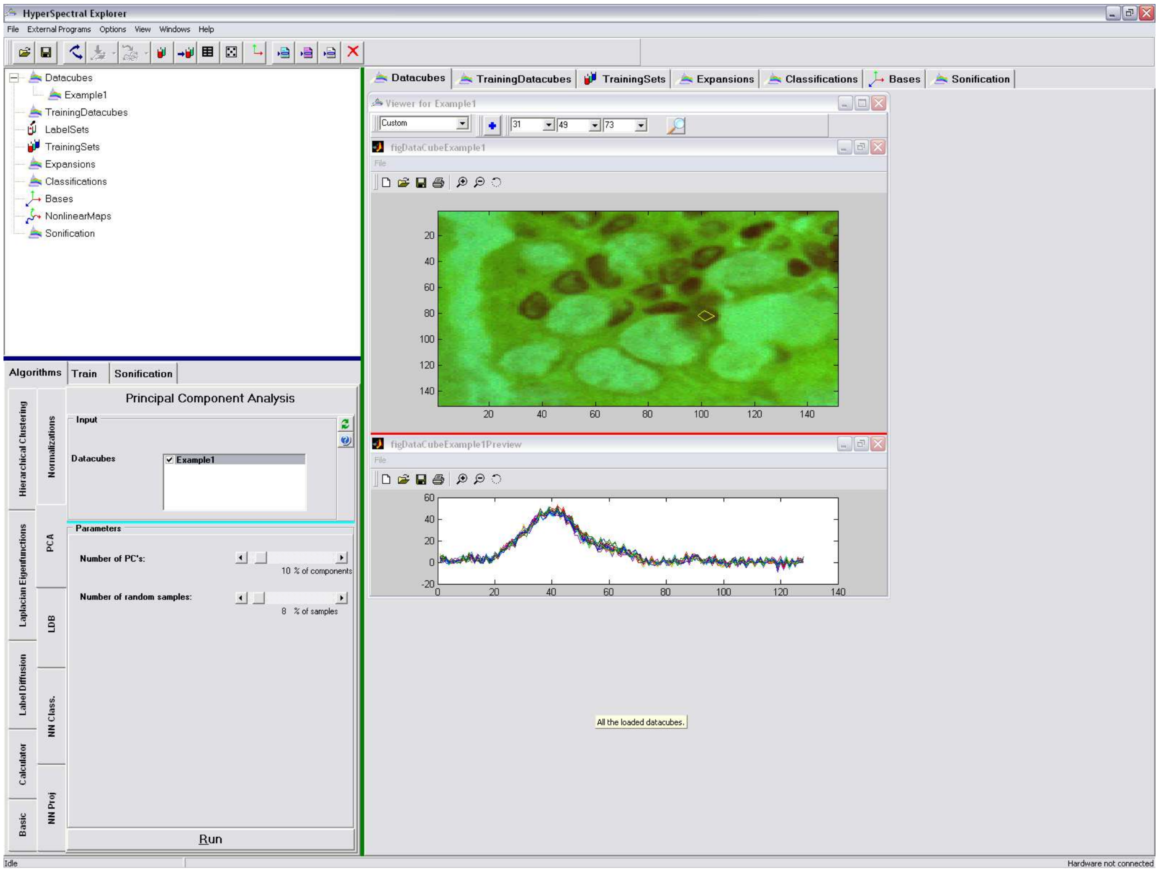
As an exercise, you can reproduce the image above by setting the parameters in the data cube browser and selecting the region of the data cube indicated above.

7.iii Running your first algorithm

Let us go through some manipulations of the dataset.

First of all, we observe that the dimensionality of the data set is quite high: namely, 128. However, these spectra are noisy and, without noise, we expect them to be smooth, hence intrinsically lower dimensional. We thus decide to compute the principal components, and to project onto the top few principal components. In order to do this, we select the algorithm tab in the *Algorithm Browser* and switch to the tab named “PCA”.
If the currently loaded datacube does not appear in the list of datacubes, press the refresh button at the right of the datacube list.
Then:

- select “Example1” in the list of input datacubes,
- move the slidebar corresponding to the “Number of PC's” to 10%, which means we want to compute only 10% of 128=12 principal components,
- move the slidebar corresponding to “Number of random samples” to 8%, which means that the principal components will be computed on a random subset of the spectra only, of size 8% of 151*151 spectra which is approximately 1800 spectra out of about 22500. We expect this to be rather accurate because the structure of the spectra does not seem that complicated. The screen should now more or less look like this:



Principal Component Analysis

Input

Datacubes ☑ Example1

Parameters

Number of PC's: 10 % of components

Number of random samples: 8 % of samples

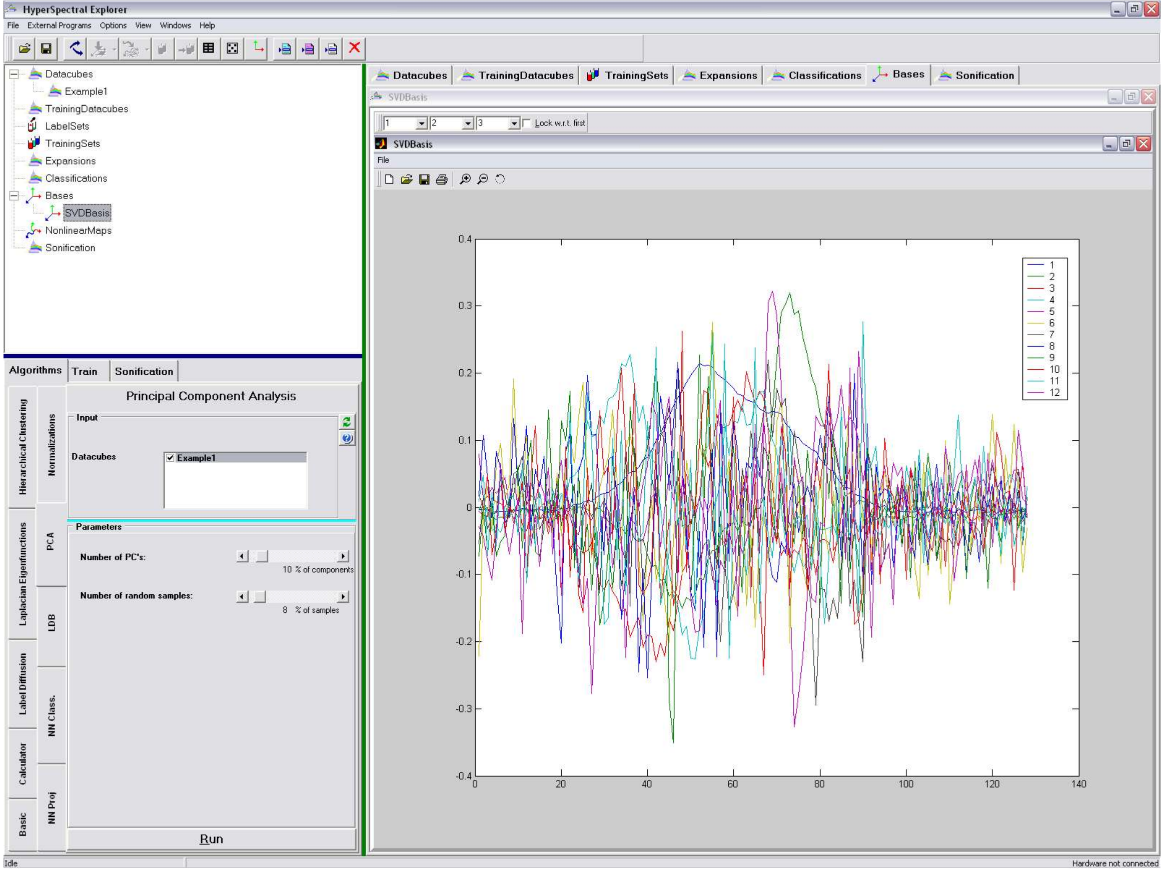
All the loaded datacubes.

Now hit the **Run** button at the bottom of the algorithm page.

You will be asked for a name for the variable being created by the algorithm, which is going to be an orthonormal set of vectors. We are going to call it “SVDBasis”. After typing the name of the variable, click Ok.

After running, the algorithm will create a variable called “SVDBasis”, in the group “Bases”.

Double click on “SVDBasis” in the *Object Tree*, and you will see the basis vectors displayed in the “Bases” page view:



Principal Component Analysis

Input

Datacubes ☑ Example1

Parameters

Number of PC's: 10 % of components

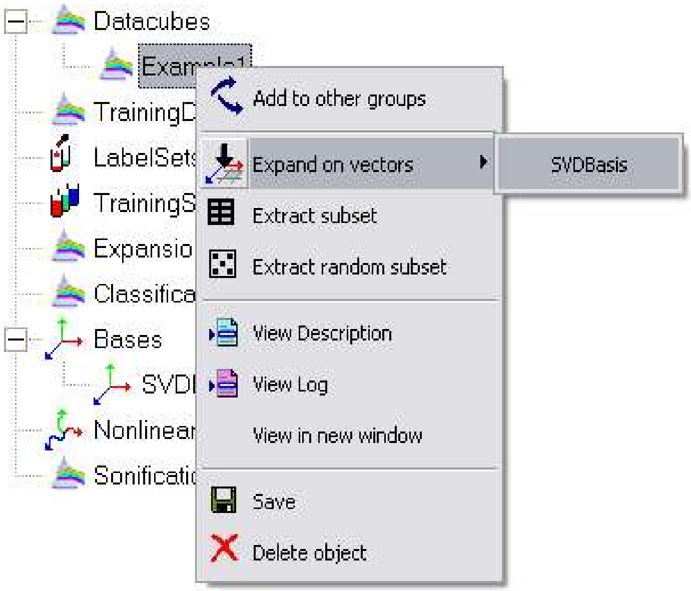
Number of random samples: 8 % of samples

We have 12 basis vectors, as you can see from the picture. You can also select SVDBasis, right click and select “View Description” to obtain information about this variable.

7.iv Linear projections

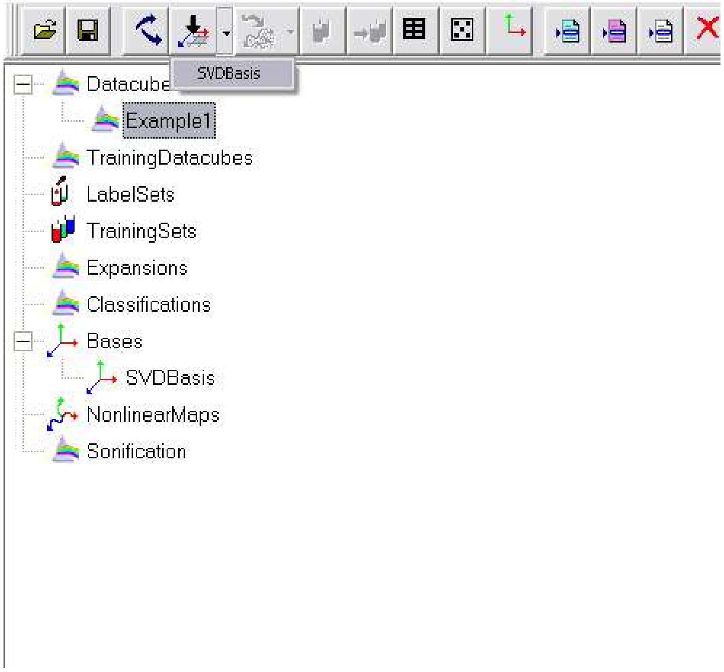
Let us go through some manipulations of the dataset.

Now we want to project the original data cube onto this basis, thus reducing the dimensionality from 128 to 12. This can be done in two equivalent ways, and in general all the actions on any object can be performed analgously in these two ways.



Using popup menus: select Example1 int the *Object Tree* and right-click with the mouse.

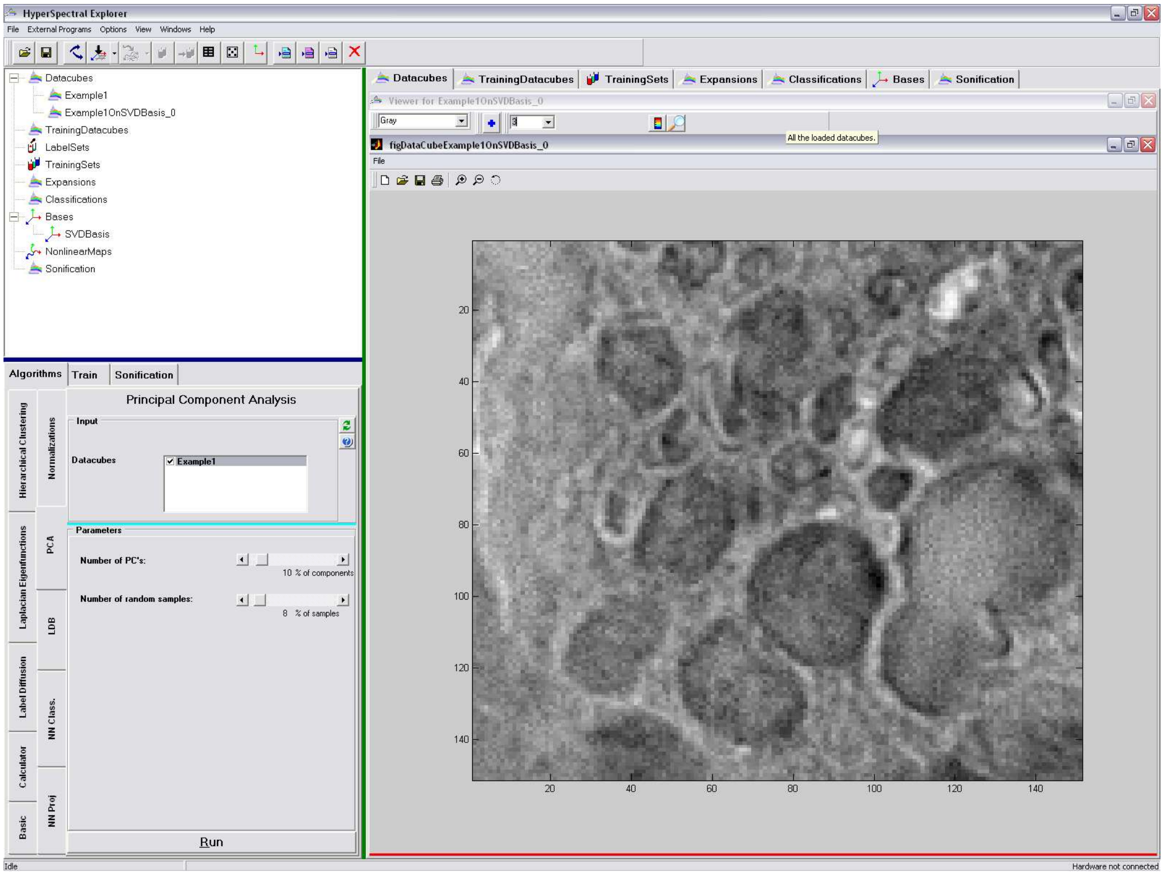
A popup menu like in the picture will appear. Select “Expand on vectors” and then “SVDBasis” in the submenu.



Using the toolbar buttons: select Example1 int the *Object Tree* and then click on the button “Expand on basis” (you can see the hints for each button simply by resting the mouse on the button for a fraction of a second): the list of available bases to project onto will drop down: click on “SVDBasis”.

Accept the choice of name for the new variable.

The algorithm will quickly run. Double-click on “Example1OnSVDBasis_0” to see a browser for the result, or simply switch to the Datacube Page.

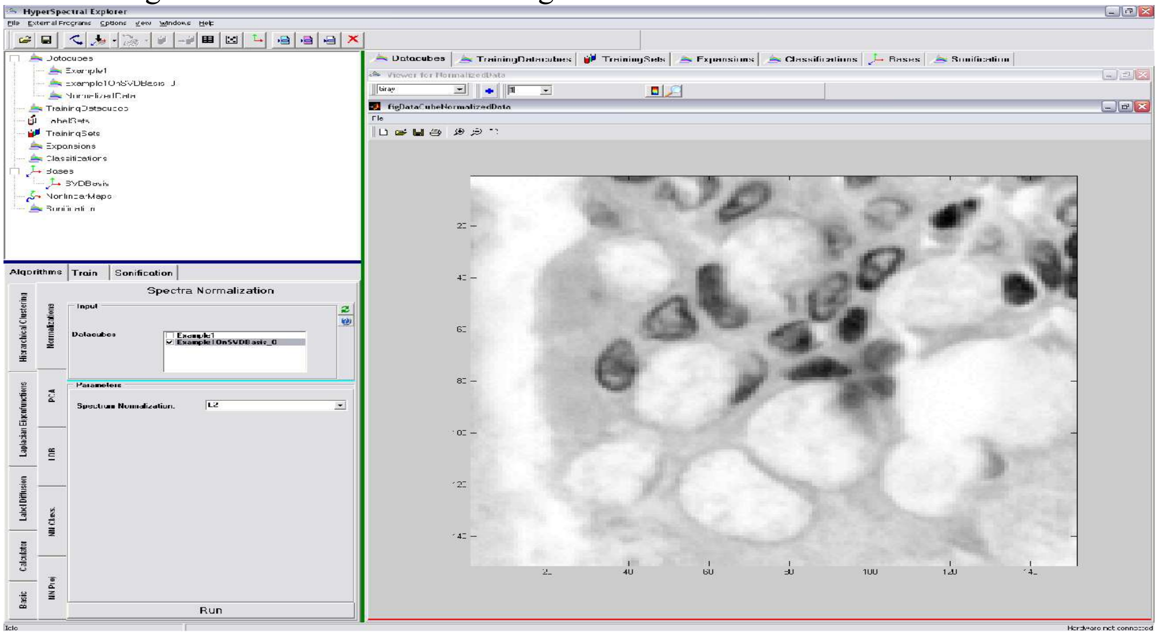


Number of PC's: 10 % of components

Number of random samples: 8 % of samples

Browse through the various layers of the cube: there are 12 of them, one per principal component computed, since each layer is the projection of the data cube onto the corresponding principal component. Observe for example that the first layer essentially represents the projection onto light-intensity (the variable with greatest variable in the data cube), the other represent finer and finer variables.

We can now project this new variable on the unit sphere: select the “Normalization” algorithm tab, select “Example1OnSVDBasis_0” as input datacube, select L2 in the “Spectrum Normalization” dropbox, and click the Run button.
Call “NormalizedData” the new variable when prompted for a name.
The resulting data cube will look something like this:



Datacubes
□ Example1
☑ Example1OnSVDBasis_0

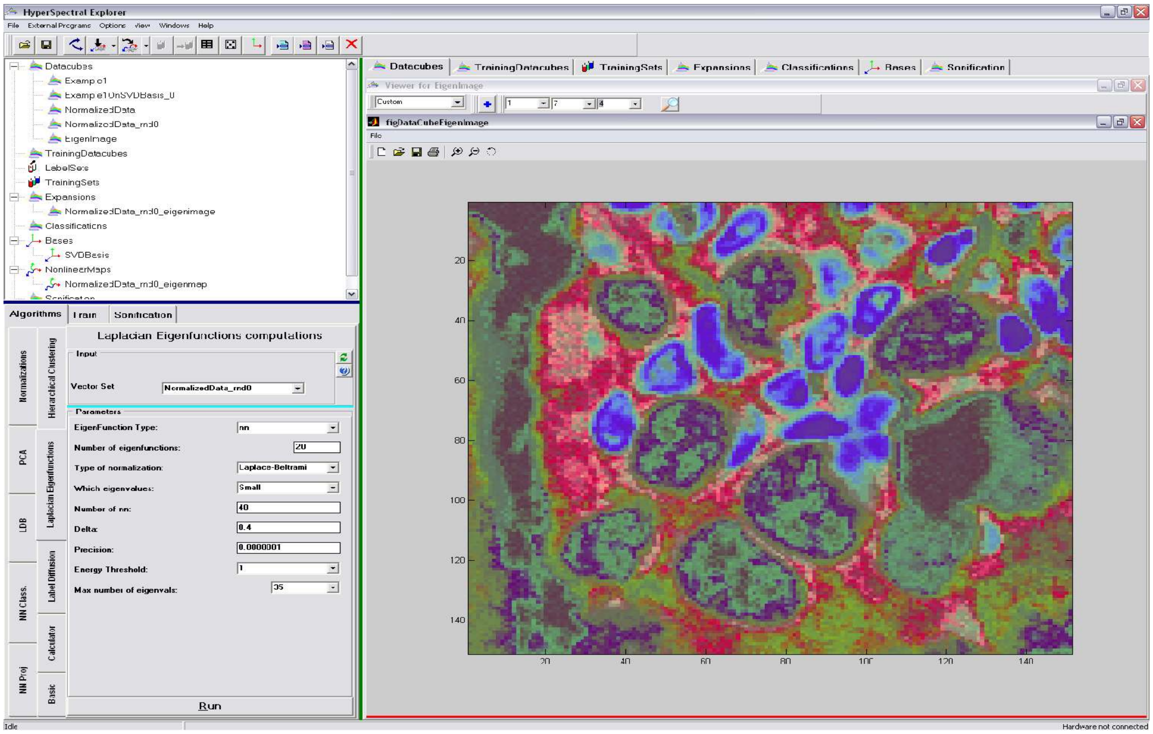
Spectrum Normalization: L2

7.v Nonlinear maps and projections

Suppose now we would like to find a good parametrization for the spectra in this data cube, by using a nonlinear dimension reduction tool such as the Laplacian eigenfunctions.
The dataset is quite large, and while feasible on a standard PC, we would like to do a quick computation, maybe paying a little in terms of precision. So we would like to extract a relatively small random subset of “NormalizedData”, compute the Laplacian eigenfunctions on this subset, and then extend them to the whole set.
In order to do this, select NormalizedData in the Object Tree.
Either right-clicking and select “Extract random subset”, or click the corresponding button on the toolbar. Select a random subset size of about 1000-1500 samples, and accept the default name “NormalizedData_rnd0” for the new variable.

Go to the “Laplacian Eigenfunctions” algorithm tab.
Choose “NormalizedData_rnd0” as input vector set.
Leave all the parameters as set by default, but change the parameter “Delta” to 0.4. This is the width of the kernel used, since all the points in “NormalizedData” are on the unit sphere, we want to localize the kernel at a scale smaller than the unit radius of that sphere.
Click the Run button, and choose “NormalizedData_rnd0_eigenmap” and “NormalizedData_rnd0_eigenimage” as names for the output variables.
The Laplacian algorithm will create a Nonlinear map, that can now be applied (extended) to any dataset of the correct dimensionality, and also the result of applying this nonlinear map to the original data “NormalizedData_rnd0” which was input to the algorithm: this is a new variable which is automatically put in the “Expansions” group.

Now we would like to apply (extend) the whole map to the whole dataset. We select “NormalizedData”, then right-click and select “Apply nonlinear map” and “NormalizedData_rnd0_eigenmap” in the submenu. Call the new variable “EigenImage”.
A datacube name “EigenImage” is created, the k-th layer representing the value of the k-th eigenfunction evaluated on the spectrum of each pixel. Browse through the datacube to get a feeling for this result.



Laplacian Eigenfunctions computations

Input

Vector Set: NormalizedData_rnd0

Parameters

EigenFunction Type: nn
Number of eigenfunctions: 20
Type of normalization: Laplace-Beltrami
Which eigenvalues: Small
Number of nn: 40
Delta: 0.4
Precision: 0.0000001
Energy Threshold: 1
Max number of eigenvals: 35

7.vi Supervised algorithms: LDB

All the algorithms we have been running till now were unsupervised, in the sense that no training set nor classes to be learnt were sought nor were fed into the algorithms. When one knows which classes he would like to classify and is able to indentify members of that class, those can be used for training purposes, and algorithms can seek which variables allow for good predictions of the desired classes.

We consider here the example of the supervised algorithm called Local Discriminant Bases.

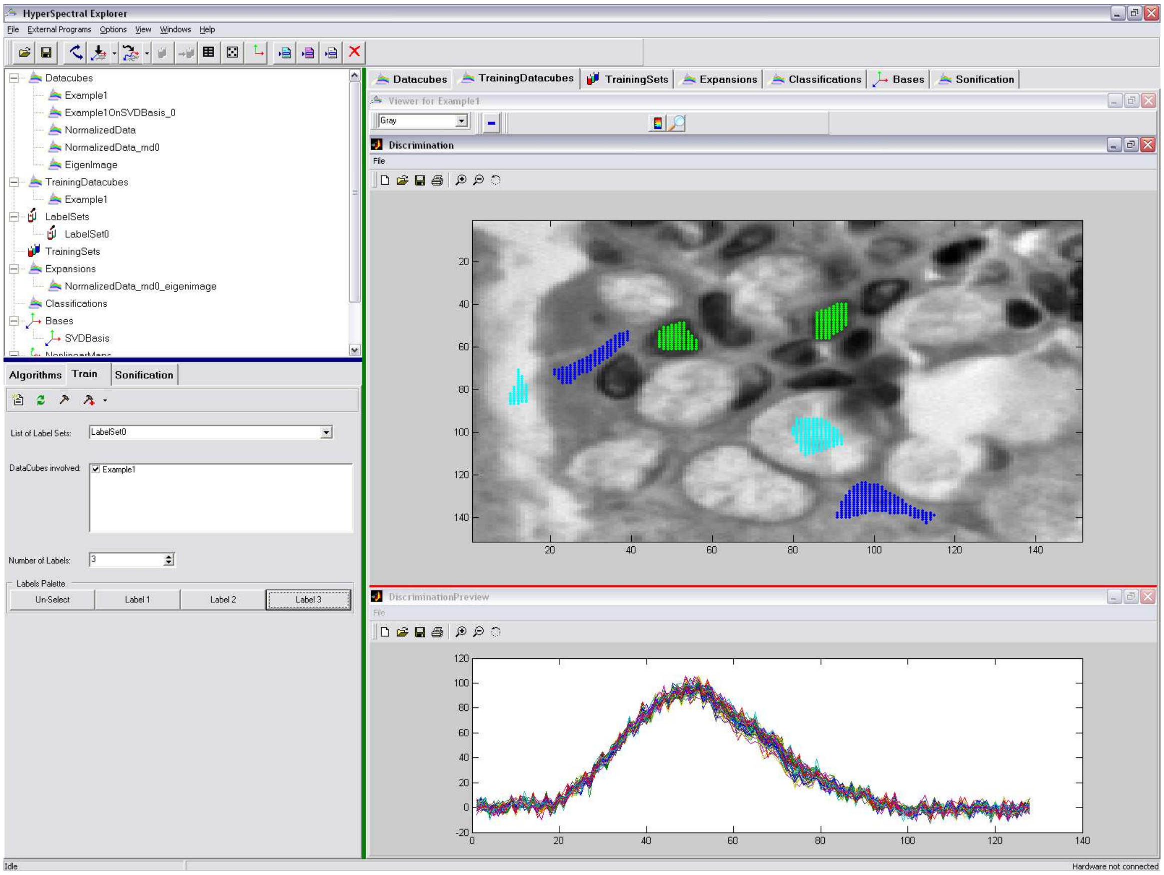
The first step is to construct a training set. To do so, we need to specify one or more data cubes to be used for training purposes. We can pick our original datacube “Example1” and add it to the “TrainingDatacubes” group. To this, we can select “Example1”, right-click and select “Add to groups” (the same result can be obtained, as the user can probably guess by now, by clicking on the “Add to groups” button in the toolbar). Then we check the “Training Datacubes” group and click Ok. “Example1” gets added to the “TrainingDatacubes” group.

We now select the “Train” page, a tab side by side with the “Algorithms” tab. Now we want to create a new training set, ie a set of vectors with corresponding labels. The software distingushes between two concepts: *'label set'* and *'training set'*. The first one denotes is a collection of labels associated to a subset of spectra from one or multiple *Training datacubes*, the second one denotes a set of labelled vectors, however obtained. Of course a supervised algorithm will act only on a training set. This is typically obtained within the HyperSpectral Explorer by creating a *'label set'* and then *'building'* from it a training set. Let us go through this process.

At this point the software has created for us a *label set* called “LabelSet0”, which allows to put labels on points in the datacube “Example1” only. If we had other datacubes in the “TrainingDatacubes” group we would be able to add more of them to the pool available to this label set. The user can also change the number of labels in the label set. For the moment, 3 is the default and is good for our purposes.

Let us build the label set as follows. Select a region from the datacube as represented in the picture below (green is label 1, blue is label 2, cyan is label 3), and label it accordingly by clicking on the corresponding label button in the train page. If a selection is wrong, simply select a region that is wrong and click the “Un-Select” button to erase any selections in that region.

Try to select more or less the regions as in the picture below. These selections are motivated by knowledge from pathology: class 1 essentially represents nuclei, class 2 cytoplasma, and class 3 is glass or other not very interesting stuff.

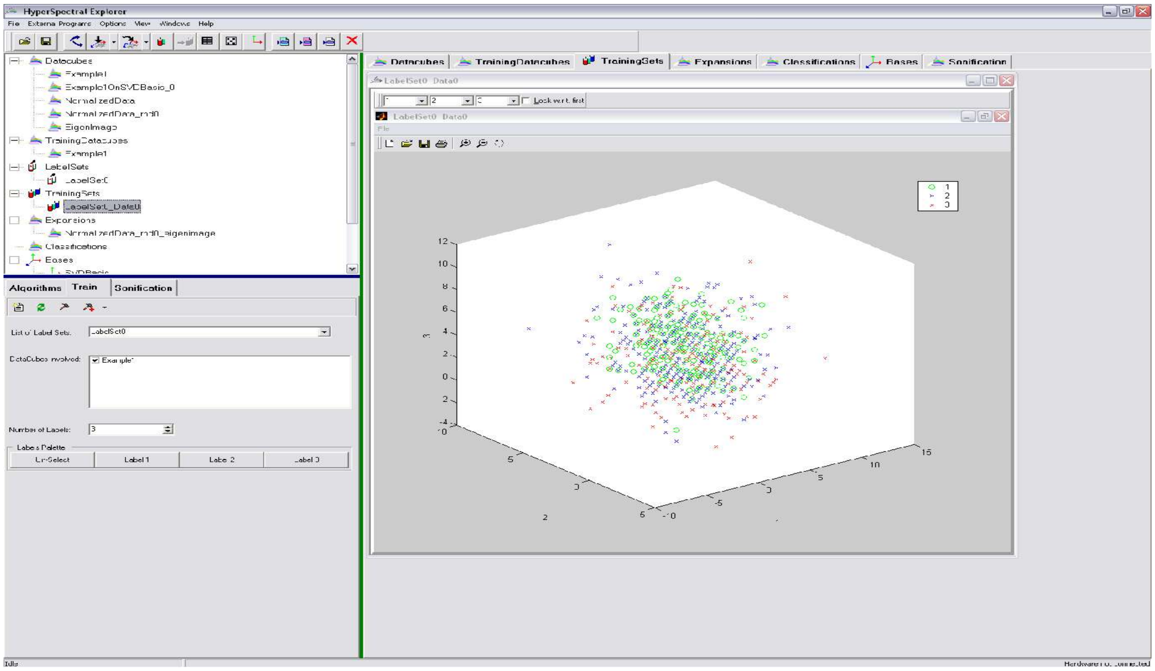


List of Label Sets: LabelSet0

DataCubes involved: ☑ Example1

Number of Labels: 3

At this point we have the desired label set. To build a training set, ie an actual set of vectors with labels, press the "Build Training Set" button, which is hammer-shaped, and the third button from the left. A TrainingSet called "LabelSet0_Data0" is created in the "TrainingSets" group. By double-clicking on it one gets a picture similar to this:



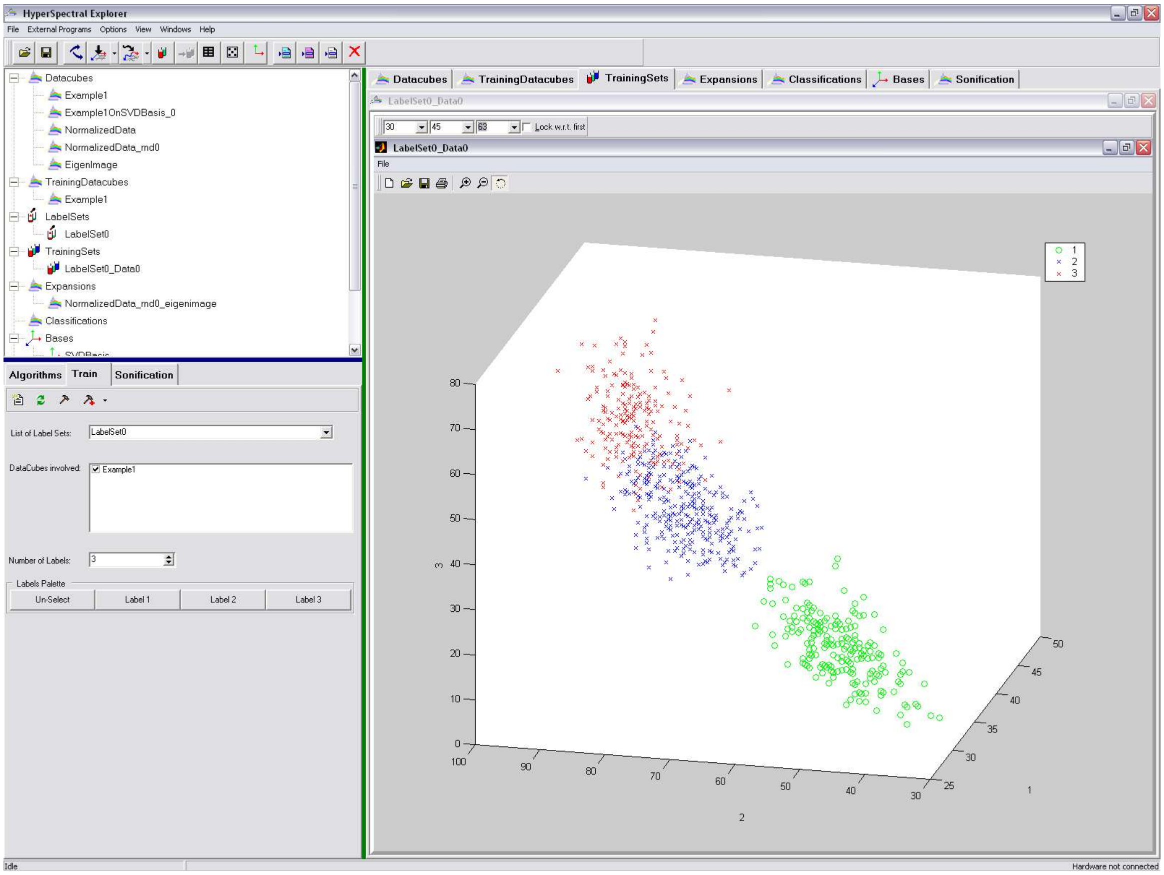
List of Label Sets: LabelSet0

DataCubes involved: ☑ Example1

Number of Labels: 3

The color of the points represents the class., and a legend at the top of the figure is a legend for the color-class correspondence. The points of different classes appear all mixed up. But in which coordinates are we looking at them? Well, these coordinates are the bottom coordinates of the spectrum, which are quite noisy, so it is not a surprise we cannot see anything. Play a little with changing the coordinates. After browsing through various coordinates and by rotating the axes a little, one could get a pictures like the following:

which looks of course much more promising towards classifying points from the different classes. Of course one has to get quite lucky, in general, to find such good coordinates. Local Discriminant Bases is able to find coordinates such that the projection onto them best preserves the discrimination between the classes. Each coordinate selected by the algorithm is in general a complicated linear superposition of the original coordinates, so it is much more flexible of what we just tried, which was to select just 3 among the original coordinates.



List of Label Sets: LabelSet0

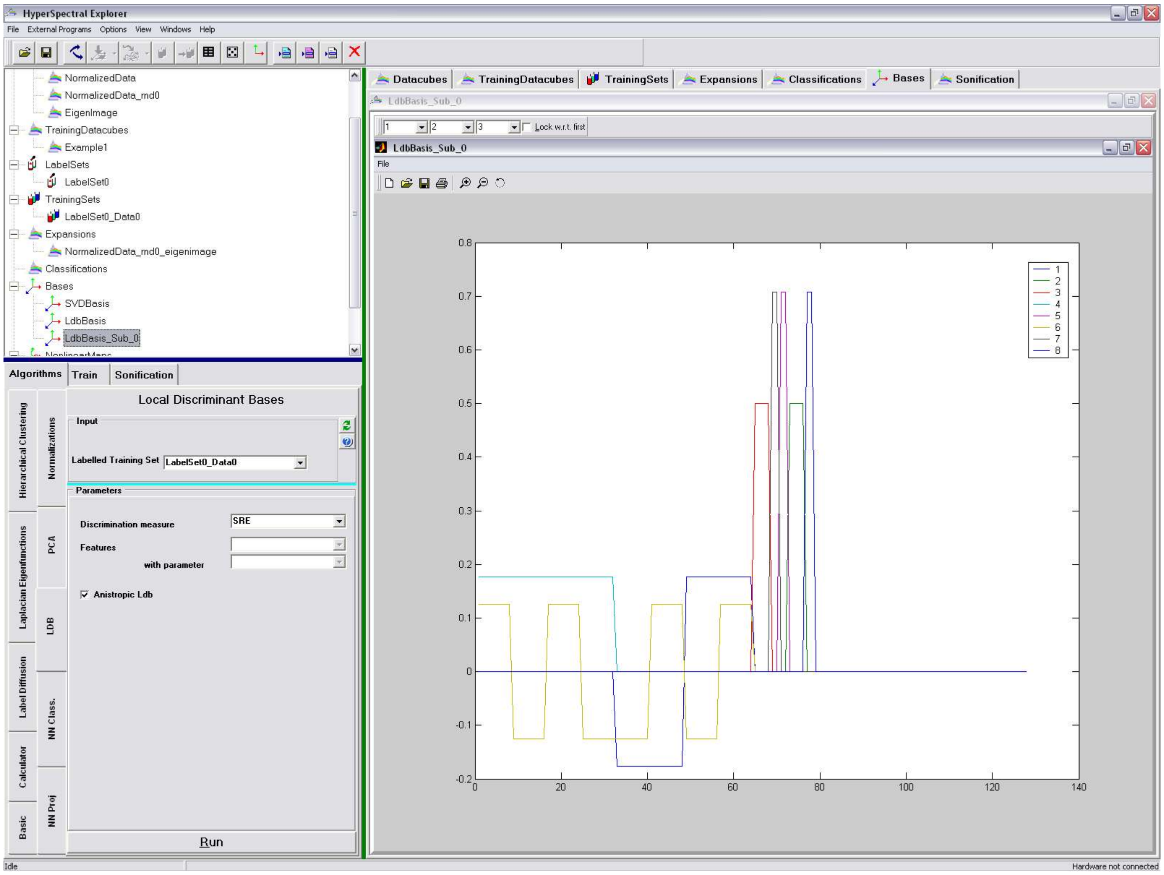
DataCubes involved: ☑ Example1

Number of Labels: 3

30 45 63 □ Lock w.r.t. first

So let's go back to the “Algorithm” tab and select LDB. Let's keep the default parameters (pretty good for rather general problems), and click Run. Let's call “LdbBasis” the output of the algorithm. When the algorithm finishes running, we have a new basis, of 128 vectors, ordered by “discrimination power”. Let's select only the top 8 vectors, say, by selecting “LdbBasis”, right-clicking and selecting “Extract Basis Subset”, and ticking only the top 8 vectors. Double-clicking on “LdbBasis_Sub_0” reveals the discrimination vectors found:

The vector found during your experiment may be different, since they depend strictly on the training set you chose.



Local Discriminant Bases

Input

Labelled Training Set: LabelSet0_Data0

Parameters

Discrimination measure: SRE

Features:

with parameter:

☑ Anistropic Ldb

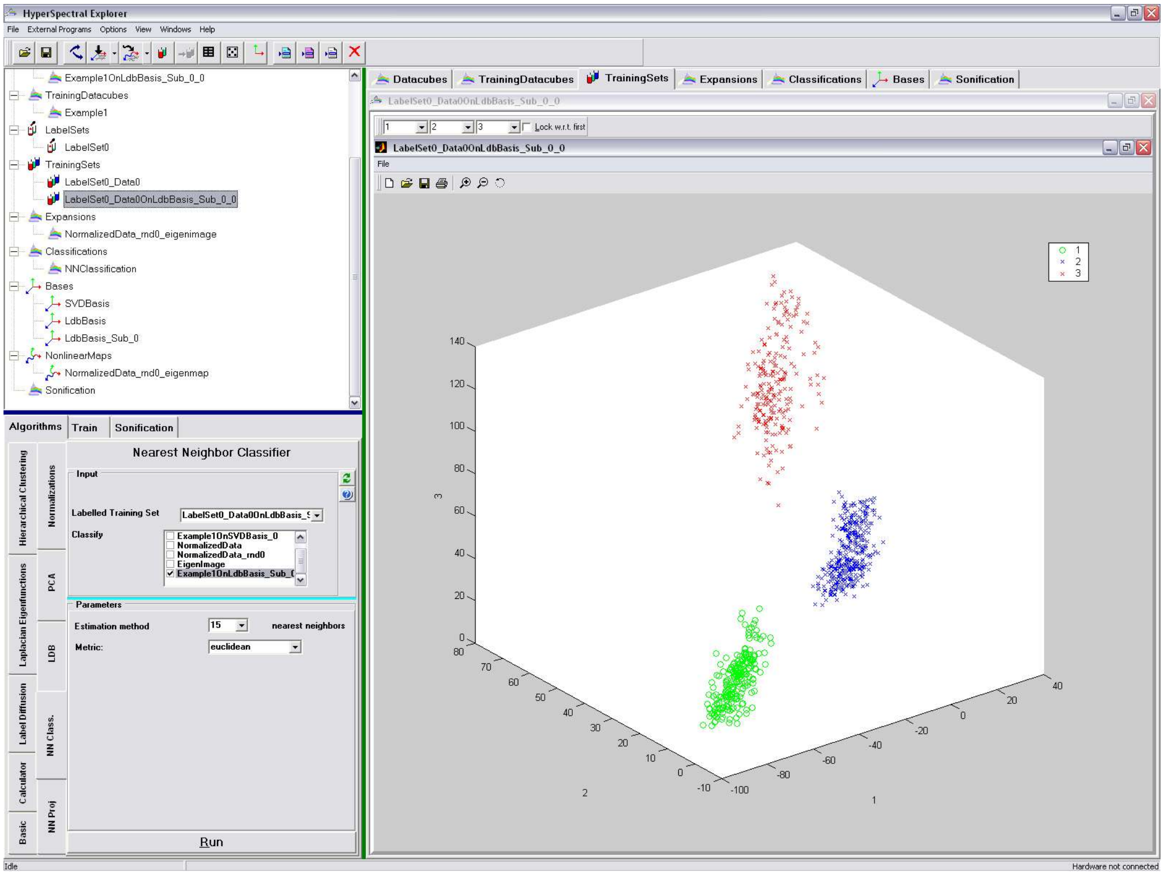
Let's project the original data cube “Example1” onto this subset of the LdbBasis (select “Example1”, right-click, select “Expand on Vectors”, then LdbBasis_Sub_0) thus getting a data cube called “Example1OnLdbBasis_Sub_0_0”. It is instructive to browse through it.

7.vii Supervised algorithms: Nearest Neighbors

We can view “Example1OnLdbBasis_Sub_0_0” as intelligently dimensionally reduced version of “Example1”, where the dimensionality reduction was training set driven, for the purpose of classification, in constrast with the “blind” principal components analysis which compresses the data without any classification goal or information.

At this point we can run a nearest-neighbor classifier on the projection. In high dimension it would be essentially bound to fail because of the high-dimensionality combined with noise, but in lower dimension the data is less noisy, because of projection onto smooth functions, and at the same time the LDB vector try not to lose too much discriminant information.

So we first project the training set “LabelSet0_Data0” onto the top LDB vectors LdbBasis_Sub_0 (select “LabelSet0_Data0”, right-click, select “Expand on Basis”, select LdbBasis_Sub_0). We get “LabelSet0_Data0OnLdbBasis_Sub_0_0”. Double-click on it to see how well the coordinates found by LDB separate the classes of the training samples:



Nearest Neighbor Classifier

Input

Labelled Training Set: LabelSet0_Data0OnLdbBasis_S

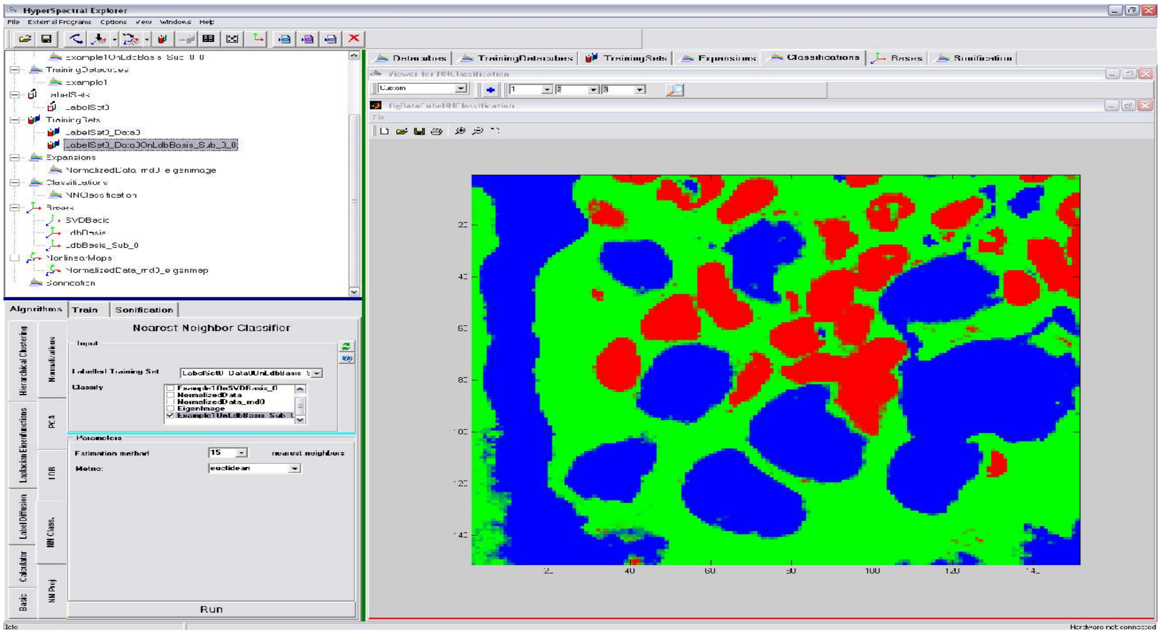
Classify:
- ☐ Example1OnSVDBasis_0
- ☐ NormalizedData
- ☐ NormalizedData_rnd0
- ☐ EigenImage
- ☑ Example1OnLdbBasis_Sub_0

Parameters

Estimation method: 15 nearest neighbors

Metric: euclidean

Then we go to the “NN Class.” algorithm. Select “LabelSet0_Data0OnLdbBasis_Sub_0_0” and “Example1OnLdbBasis_Sub_0_0”, and leave the other parameters as default. Click Run. Call the output “NNClassification”. A data cube will be created, the k-th layer representing the probability of each pixel to be in class k, and browse through or combine the layers:



Nearest Neighbor Classifier

Input

Labelled Training Set: LabelSet0_Data0OnLdbBasis_S

Classify:

- □ Example1OnSVDBasis_0
- □ NormalizedData
- □ NormalizedData_md0
- □ EigenImage
- ☑ Example1OnLdbBasis_Sub_0

Parameters

Estimation method: 15 nearest neighbors

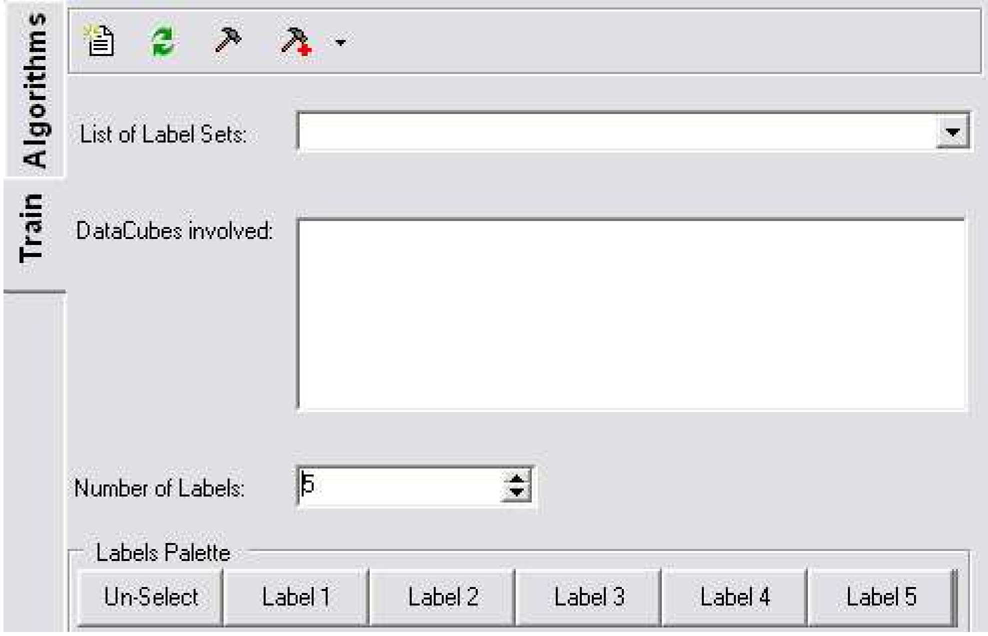
Metric: euclidean

Run

8 Building Label Sets and Training Sets

8.i Layout

This is how the Train page may look



List of Label Sets:

DataCubes involved:

Number of Labels: 5

8.ii Definitions

A *Label Set* is simply a set of labels attached to corresponding vectors in one or more datacubes. It is an enseble of graphical objects, not a set of vectors. A *Label Set* is defined

by:
- the set of Training Datacubes from which labelled samples can be selected,
- the set of Labels that can be used (usually numbered from 1 to L)

A *Label Set* differs from a *Training Set* in that a *Training Set* is the set of vectors (living in some N-dimensional vector space) with the labels attached, which can then be input to any supervised classification algorithm. After defining the *Label Set* the user can *Build* a *Training Set* from the *Label Set* by selecting the *Build* button from the Train toolbar at the top of the Train page.

8.iii User Interface

Situated at the top of the Train page is a **toolbar**, that contains four icons:
1) *New label* set: creates a new label set, adding it to the list of available Label Sets.
2) *Refresh* the list of available datacubes, in case some datacubes have been recently loaded/moved and don't appear in the list of available datacubes.
3) *Build* a training set out of Label Set.
4) *Add* a Label Set to an existing Training Set.

Below the toolbar, there is a **combobox** listing the existing *Label Sets*.

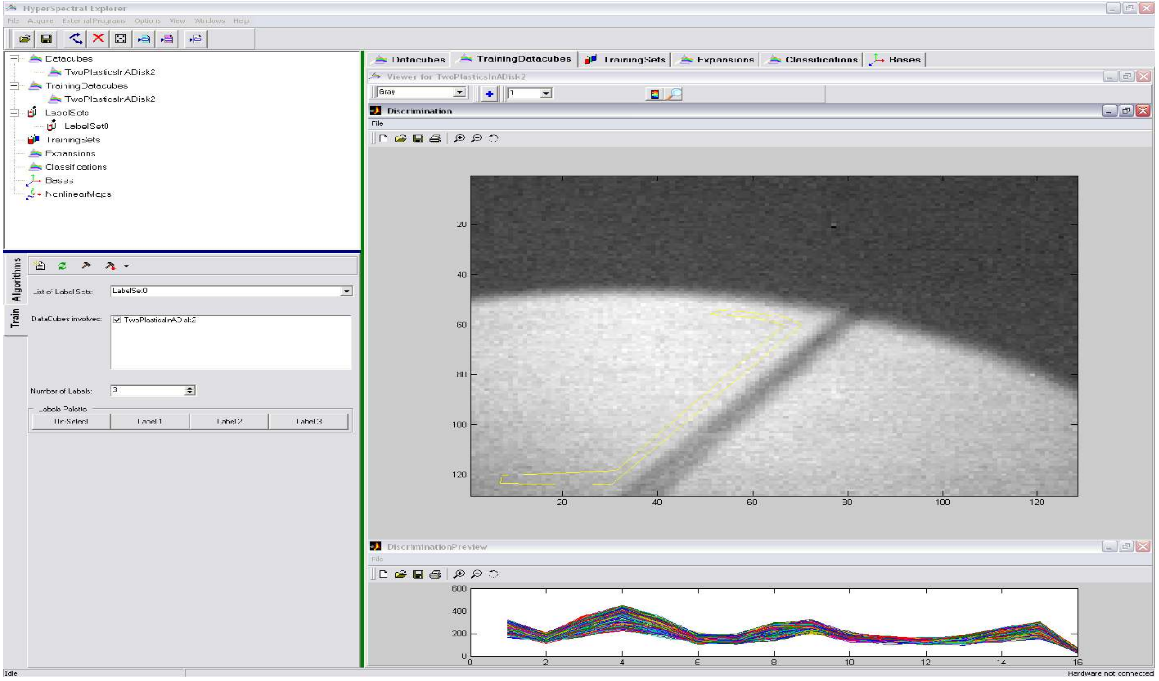
Below the combobox, there is the **check list** of available Training Datacubes. This should contain all the Training Datacubes showed in the Object TreeView, if it does not, the user can force an update by clicking on the Refresh button in the toolbar. Each Training Datacube can or not partecipate in a *Label Set*, and this is determined by whether it is checked or not. If a Training Datacube is not checked, the user will not be allowed to add labelled samples from that Training Datacube. Otherwise, selected labelled samples from that Training Datacube can be added. If a Training Datacube is un-checked after samples from it where added to the *Label Set*, those samples will be removed from the *Label Set*.

Below the list of Training Datacubes is a control to select the **number of labels**. This controls how many different labels can be assigned to the points. When then number of labels is decreased, already assigned labels are not erased, but they will not be considered when building a training set.

8.iv Constructing a Label Set

1. Add/move the datacubes that you want to use in the Label Set to the Training Datacubes group.
2. If a *Label Set* does not exist, create a new one by pressing the *New* button in the *Train toolbar*.
3. In the check list of *Datacubes involved* check the Training Datacubes that will partecipate in the Label Set (can be changed at any later time). If the list does not contain all the available Training Datacubes, click the *Refresh* icon in the toolbar to force an update in the list.
4. Select the number of labels that will be used in the Label Set.
5. The screen may look like this:

6. To add sample to a certain class, click on the Datacube Browser: the mouse cursor will become a crosshair, and every click will define a vertex of the polygonal region you are selecting. When you are finished selecting the region, double-click, or press Enter.



List of Label Sets: LabelSet0

DataCubes involved: ☑ TwoPlasticsInADisk2

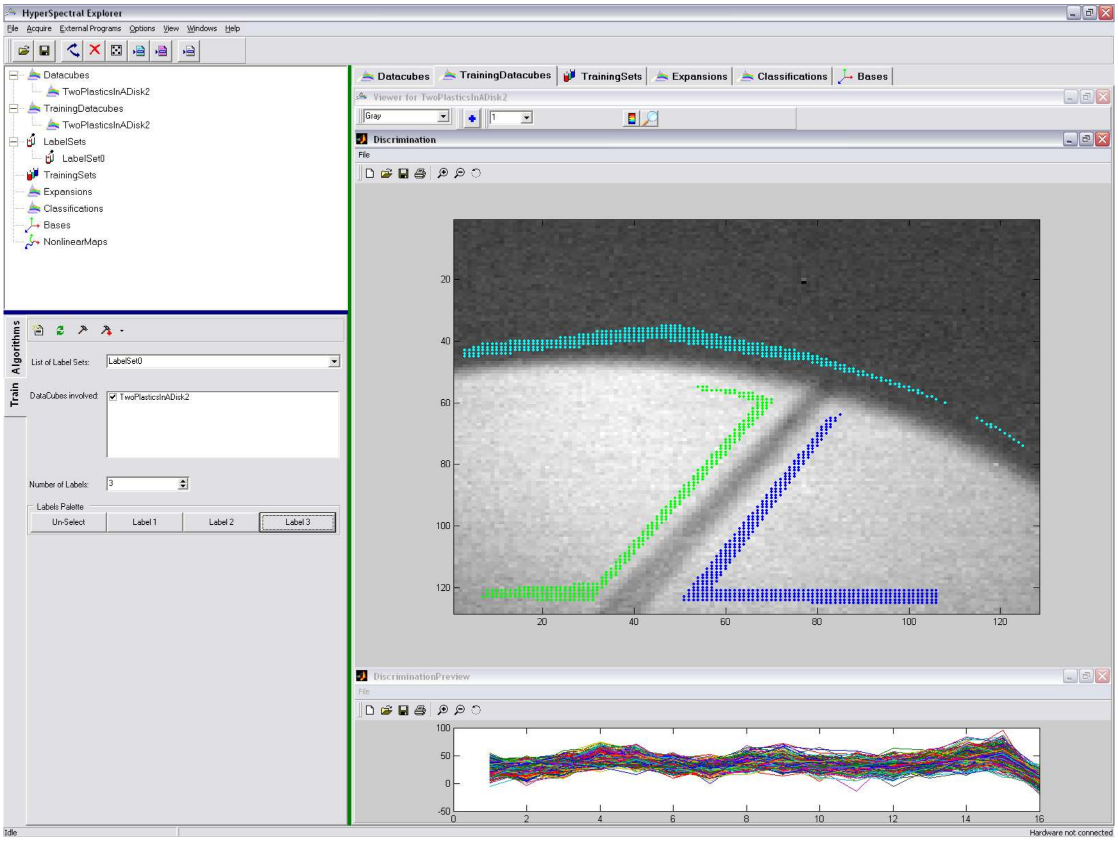
Number of Labels: 3

Labels Palette

Un-Select | Label 1 | Label 2 | Label 3

7. To add the selected samples to the Label Set with a certain label, simply press the button in the *Labels Palette* on the Training page corresponding to the label you want to assign. The selected region will be colored with a color corresponding to the label.

8. If you are unhappy by any part of your selection, simply select a region that you want to remove, and click the *Un-select* button in the *Labels Palette*: all the samples in the selected region, regardless of whichever class they were assigned to, will be removed from the Label Set.

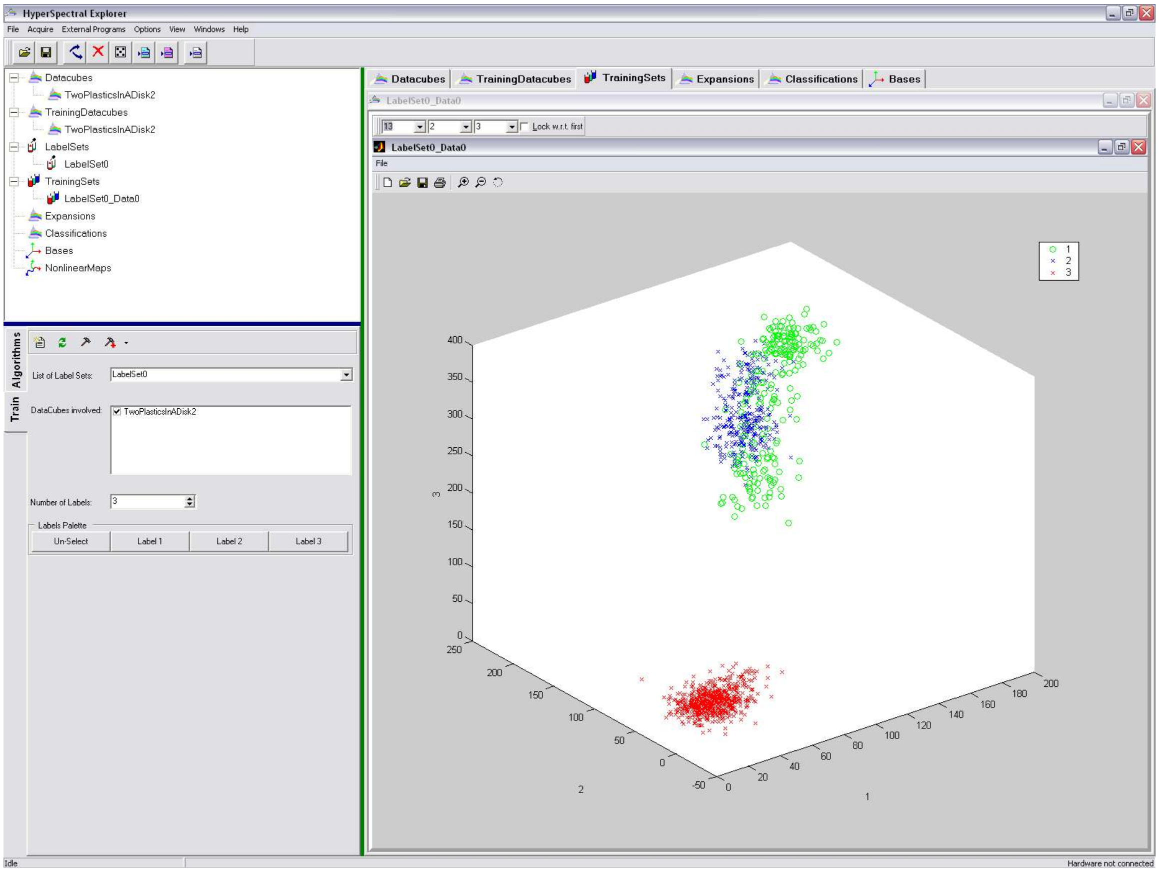


List of Label Sets: LabelSet0

DataCubes involved: ☑ TwoPlasticsInADisk2

Number of Labels: 3

9. When you are happy with your selections, press the *Build* button in the Train page toolbar to build a *Training Set*. The training set you will create will be added to the group of *Training Sets* and displayed in the corresponding page of the Object PageViewer.



List of Label Sets: LabelSet0

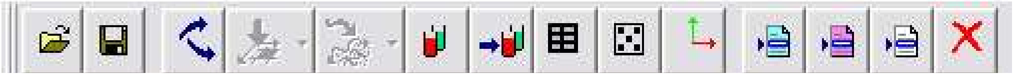
DataCubes involved: ☑ TwoPlasticsInADisk2

Number of Labels: 3

9 The Main Toolbar

9.i Overview

The main toolbar contains buttons to perform most of the available actions on the objects inside HyperSpectral Explorer.



Here is a description of their functionality, from left to right.

9.ii Open

Opens a Matlab, HSE or pnd file. The same as the command accessible from the main menu: File-Open.

9.iii Save

The same as the command accessible from the main menu: File-Save.

9.iv Add to groups

Used to copy an object to a different group. For example a daata cube in the "Datacubes" group could be moved to the "Training Datacubes" group.

9.v Expand on a basis

Used to project a given data cube or set of vectors onto a basis object. Select the object to be projected in the Object TreeView, and then the basis to project onto from the drop-down menu of the toolbar.

9.vi Apply nonlinear map

Used to apply to a nonlinear map object to a data cube or set of vectors. Select in the Object TreeView the object the map should be applied to, and then the nonlinear map to apply from the drop-down menu of the toolbar.

9.vii Extract labelled subset

Extracts a subset of a training set or in general of a set of vectors with labels, based on a choice of labels. Very useful if one wants to study further one class obtained through some classifier: this class can be extracted using this feature and then further processed with any algorithm.

9.viii Classification to training set

Converts a classification object into a training set, i.e. a set of vectors with the labels assigned by the classifications.

9.ix Extract regular subset

Allows to extract a subset from a data cube in a regular fashion, allowing to downsample the set in a geometrically sensible way. For example one can discard all even rows and columns in the x and y coordinates and keep only one sample every 8 in the spectral dimension.

9.x Extract random subset

Extracts a random subset from a data cube. Can be used to downsample randomly a data set, run a learning algorithm on the random subset (due to memory and/or time constraint), and then, if possible, extend the results from the random subset to the whole set. This is tipically done, for example, in order to compute the Laplacian eigenfunctions (see the example in this user's guide).

9.xi Extract basis subset

Allows the user to pick only a few vectors out of the collection of vectors in a basis objects. For example, after computing the principal components, one may want to keep only the few top vectors. Same would happen for LDB basis vectors.

9.xii View variable description

Shows a description of the currently selected object in the Object TreeView.

9.xiii View variable log

Shows the log of the actions performed on the selected object in the Object TreeView.

9.xiv View application log

Show the log of the actions performed by the whole application.

9.xv Delete selected object

Deletes the object selected in the Object TreeView.

10 Files and Names of Variables

10.i Names of variables

Names of variables can only contain letters, numbers and underscores, and cannot begin by a number. In particular, spaces, deshes, columns, dots, commas etc...are not allowed in any variable name.

10.ii File names

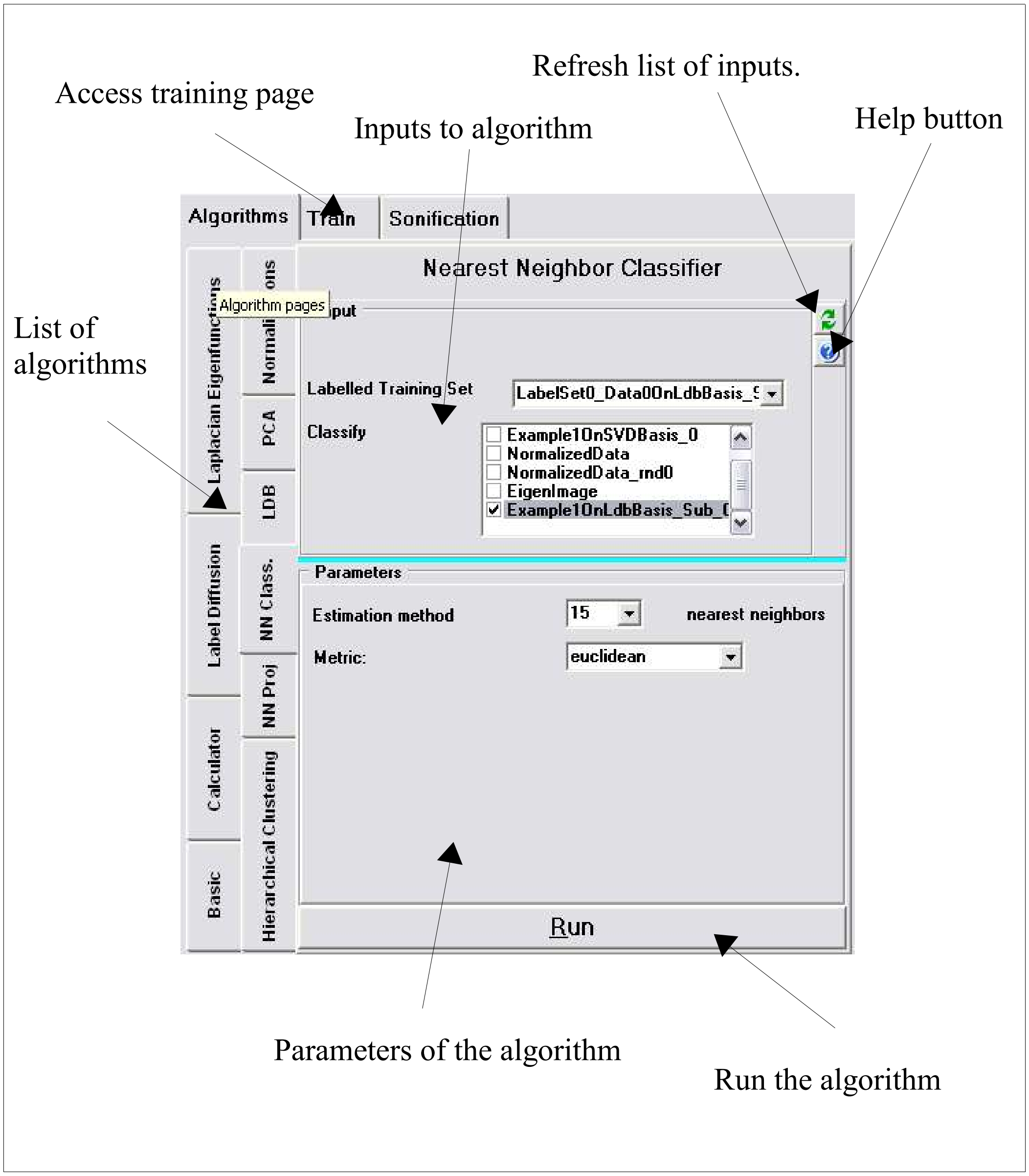
File names are subject to the same rules as variable names. Note that this poses considerable restrictions on filenames with respect to the Universal Naming Conventions standards.

11 Algorithms

11.i Layout and getting help

This is how the Algorithm page may look like. Differences from the look on your system are due to different configuration, installation or system settings.

The Algorithm page ontains itself several pages, one per available algorithms.



Labelled Training Set: LabelSet0_Data0OnLdbBasis_S

Classify:
□ Example1OnSVDBasis_0
□ NormalizedData
□ NormalizedData_rnd0
□ EigenImage
☑ Example1OnLdbBasis_Sub_0

Estimation method: 15 nearest neighbors

Metric: euclidean

Different sets of algorithm may be installed on your system.

Technical Note: Algorithms can be designed so that they plug-in quite naturally into the HSE engine, and we are considering releasing tools that will make this process accessible to third parties.

Every algorithm page has its own title, a small toolbar with a refresh and a help button, an area for input selection, an area for parameter selection, and a run button at the very bottom.

The refresh button refreshes the list of available input parameters, when this is not synchronized with the actual object list.

The help button provide algorithm specific help on what the algorithm does, on its input parameters, etc.

Example. The algorithm *Principal Components* snapshot above. There is only one input category for this algorithm, described as "datacubes". The object category from which admissible input objects can be selected is of course "Datacubes". These objects are listed (if the list is not updated, just press the refresh button), and the user can select one or more datacubes to which to apply the algorithm. The algorithm will be applied to each input datacube, in a serial fashion. The parameter section includes the two parameters to the algorithm: the number of principal components to compute and how many random samples to use for the computation. When the user presses 'Run' after selecting the input objects and the selected the desired parameters, he will be asked for the name of the (one) output object. Should the algorithm have more than one output, the names of all the outputs would be asked, one at a time. The output is a basis, and it will automatically added to the correct category when the algorithm returns it.

Inputs

The input parameters area contains a set of controls that allow the user to select which objects should be input to the algorithm. Many algorithms have just one input of one type, but in general an algorithm can have many inputs. Inputs are divided by "conceptual role" in the algorithm, inputs with the same "conceptual role" come from the same object category, and can be a single object or multiple objects from the category, depending on the algorithm.

Parameters

This section of the User Interface allows the user to select the desidered parameters for the algorithms. The role and admissible range of each parameters are detailed in the help to the algorithm, which can be obtained by pressing the help button on the algorithm's toolbar.

Outputs

Just before running, the algorithm will ask the user for the name(s) of the output variable (s). After the algorithm is run, the output objects will be added to the set of avilable objects, each in the appropriate category.

11.ii Basic manipulations

Short Description

Basic manipulations on a vector set

Detailed Description

Allows to perform various basic manipulations of a vector set or a data cube, such as a adding a constant, and thresholding from below and above. The operations are performed in the order in which they are listed.

Parameters

Label	*Description*
Absolute value	Takes the absolute value of all the vector components of each vector.
Add Constant	Adds a constant to all the vector components of each vector.
Threshold below	Sets to the specified threshold all the elements below the specified threshold.
Threshold above	Sets to the threshold all the elements above the specified threshold.

Inputs

Name	*Description*	*Multiplicity*
Vector Set	The vector set to be operated upon.	0

Outputs

Name	*Description*	*Multiplicity*
Output Vector Set	The result of the manipulation of the input vector set.	Vector Set

References

Label	*Title*

11.iii Calculator

Short Description

Calculator between data cubes.

Detailed Description

Allows various computations to be done on datacubes.

Parameters

Label	*Description*
Operation to perform	The operation to be performed on the two datasets - +:Sum of the two vector sets. - -:Difference of the two vector sets. - *:Product of the two vector sets. - /:Ratio of the two vector sets. Division by 0 yields 0..

Inputs

Name	*Description*	*Multiplicity*
Vector Set 1	The set of points for the first operand.	1
Vector Set 2	The set of points for the second operand.	1

Outputs

Name	*Description*	*Multiplicity*
Output Vector Set	The result of the operation between the two datacubes.	1

References

Label	*Title*

11.iv Unsupervised Hierarchical Clustering

Short Description

Finds clusters in dataset in an unsupervised way.

Detailed Description

Uses distance relationships between points, in a variety of metrics and with different definitions of similarities between groups of points, to find clusters in the dataset. The algorithm is randomized to allow the analysis of large datasets: one or more selections of random groups are followed by a hirarchical clustering on each group, then the resulting clusters are matched among the various sampling rounds to yield a more robust result. The algorithm is in general badly affected by high dimensionality of the data, since it based on distance computations of in general noisy data. Usually best results are obtained by running this algorithm after an initial dimension reduction

Parameters

Label	***Description***
Metric	The type of metric to use when discovering clusters. Please note that for the density estimation the standard euclidean metric is always used, no matter which type of density estimation. • euclidean:standard euclidean distance. • seuclidean:standardized Euclidean distance, each coordinate in the sum of squares is inverse weighted by the sample variance of that coordinate. • cityblock:city block (or L^\infty) distance. • mahalanobis:mahalanobis distance. • minkowski:minkowski distance with exponent 2. • cosine:one minus the cosine of the angle between the samples (treated as vectors). • correlation:one minus the sample correlation between samples (treated as sequences of values). • hamming:hamming distance, percentage of coordinates that differ. • jaccard:one minus the Jaccard coefficient, the percentage of nonzero coordinates that differ.

Linkage type	Type of linkage between points to discover clusters • single:. • complete:furthest distance. • average:average distance. • centroid:center of mass distance. N.B.: the output is meaningful only if euclidean distance is used. • ward:inner squared distance.
Number Of Classes	How many clusters to look for.
Number of Random Points	Determines how many random points per cycle are considered.
Number of Cycles	How many clustering cycles to do; each cycle is on a different random subset of points, the clusters obtained from different cycles are then matched across cycles to yield a coherent final labelling.
Estimation Method	Density estimation for the test points. If nearest neighbor, allows to select how many nearest-neighbors to consider when assigning a label to a test point. If pdf estimation, the probability density is estimated with smooth kernels around the training points. • nearest neighbor:estimate the class densities by putting a point in a class if that class is the closest one to the point, where the distance between a point and a class is the distance between the point and the closest point in the class. N.B.: uses euclidean metric for nearest neighbor, independently of the choice of the parameter <Metric> above. • decaying pdf:estimate the class densities by smooth interpolation with a decreasing radial kernel. N.B.: the norm in the definition of the dereasing kernel is euclidean, independently of the parameter <Metric>.

Inputs

Name	***Description***	***Multiplicity***
Datacubes	The datacubes to be clustered.	0

Outputs

Name	***Description***	***Multiplicity***
Classifications	These datacubes (one per input datacube) are the classifications obtained from running the algorithm. The i-th slice of each at each point is the probability of that point belonging to the i-th cluster found.	Datacubes

References

Label	***Title***
[1]	http://www.resample.com/xlminer/help/HClst/HClst_intro.htm
[2]	http://www.cs.umd.edu/hcil/multi-cluster/index.html

11.v Laplacian Eigenfunctions computations

Short Description

Computes the Laplacian Eigenfunctions of the data set.

Detailed Description

Computes the Laplacian Eigenfunctions of the data set, by vieweing the dataset as a weighted graph and computing the eigenfunctions of a version of the Laplace operator restricted to the dataset. Various type of eigenfunctions can be computed, according to the normalization type for the operator restricted to the dataset, and other parameters for its computation (e.g. number of neighbors to consider, width of the kernel, precision etc..). This is usually very useful for nonlinear dimensionality reduction, for discovering interesting parametrization and low-dimensional embeddings of the data set. Clustering algorithms are expected to perform better after the map derived from the Laplace eigenfunctions is applied to the data since cluster are in general enhanced

Parameters

Label	***Description***
EigenFunction Type	Type of eigenfunctions: nearest-neighbor or gaussian. • nn:uses (a symmetrization of) the kernel K(x,y)=1 if x is a k nearest neighbor of y, K(x,y)=0 otherwise. • gauss:uses (a symmetrization of) the kernel K(x,y)=e^(-||x-y||/delta) if x is a k nearest neighbor of y, K(x,y)=0 otherwise..
Number of Eigenfunctions to compute	How many eigenfunctions to compute. The maximum number is equal to the cardinality of the dataset, normally far fewer (tens or a few hundreds, independently of the cardinality of the dataset, and only depended on some notion of complexity of the data) are computed.

Type of Normalization	How the operator on the dataset is normalized. In the following D(x,x) is the sum of K(x,y) for all y's, D(x,z)=0 if z is not x, while W(x,y) is the weights between x and y (1 if <Eigenfunction type>=='nn', a weighted exponential if <Eigenfunction type>='gauss'). • Laplace-Beltrami:the operator is normalized as in graph theory, but the eigenfunctions are rescaled and are the same as the normalization D^{-1}*(D-W). The advantage of this normalization is that the problem is symmetric.. • D^{-1/2}*(D-W)*D^{-1/2}:the operator is normalized as in graph theory. The problem is symmetric, but the oeprator is not an averaging operator, and it does not coincide in general with the Laplace-Beltrami operator on a manifold.. • D^{-1}*(D-W):the operator is averaging on the set, but the problem is not symmetric. Its eigenfunctions are the same as "Laplace-Beltrami", but in that case they are computed via a symmetric problem.. • D-W:the operator is symmetric but not averaging, the density of points is not normalize out. • W:no normalization.
Which EigenValues	Which eigenvalues should be computed: small or large. • small:compute the small eigenvalues (they start from 0 up). • large:compute the large eigenvalues.
Number of nn	Number of nearest neighbors to consider in the computation of the eigenfunctions. Should be large enough especially if <EigenFunction Type> is 'gauss', but not large, especially if <EigenFunction Type> is 'nn', otherwise the graph considered is complete.
Delta	Scale of the operator. This has no effect if the Eigenfunction Type is 'nn', while it affects the width of the gaussian kernel when EigenFunction Type is 'gauss'. The gaussian kernel is written in the form e^((-||x-y||/delta)^2).
Precision	Precision of the computations when EigenFunction Type is 'gauss'.
Energy Threshold	Keep only enough EigenValues to cover the given fraction of total energy. The total energy is estimated only on the parameter "Number of EigenFunctions to compute".
Maximum number of EigenValues	Maximum number of EigenValues to compute.

Inputs

Name	***Description***	***Multiplicity***
Vector Set	The set of points on which to compute the eigenfunctions.	1

Outputs

Name	***Description***	***Multiplicity***
EigenMap	The EigenMap associated with the computed Eigenfunctions	1
EigenMap Images	The image under the Eigenmap of the input <Labelled Training Set>.	1

References

Label	***Title***
[1]	S. Lafon, R.R. Coifman, Geometric Harmonics, Tech Report, CS Dept., Yale University, 2003
[2]	S. Lafon, R.R. Coifman, Diffusion Maps and Geometric Harmonics, Tech Report, CS Dept., Yale University, 2004
[3]	R.R. Coifman, M. Maggioni, Multiresolution Analysis associated to diffusion semigroups: construction and fast algorithms, Tech Report, CS Dept., Yale University, 2004

11.vi Local Discriminant Bases

Short Description

Finds local features that well discriminate the classes as labelled in the training set.

Detailed Description

Uses fast Fourier and wavelet algorithms to look up a dictionary for feature vectors that well-discriminate between the classes in the Labelled Training Set [1]. Offers several choice of libraries of bases, based on windowed local cosines, wavelets and other filter banks.

Parameters

Label	*Description*
Discrimination Measure	Determines which discrimination measure and cost are associated with each the projection of the points onto a feature. • SRE:symmetric KL distance on the coefficients. • ARE:asymmetric KL distance on the coefficients. • SED:Square Euclidean Distance on the coefficients.
Features	Determines in which dictionary to search the feature vectors: options include various wavelet packet dictionaries and sine and cosine libraries. • Haar:Haar wavelets. • Beylkin:Beylkin wavelets. • Coiflet:Coifman wavelets (maximum number of vanishing moments). • Daubechies:Daubechies compactly supported wavelets. • Symmlet:Symmlet wavelets, which are 'maximally symmetric'. • Vaidyanathan:Vaidyanathan filters.
Anisotropic Ldb	Determines whether to use the most flexible Haar-Walsh tiling of the time-frequency plane.

Inputs

Name	*Description*	*Multiplicity*
Labelled Training Set	The set of vectors with labels to be used for training set. The vectors are all of the same dimensions, while the labels are integer numbers starting from 1.	1

Outputs

Name	***Description***	***Multiplicity***
Feature Vectors	Set of feature vectors. These are vectors in the same space as the vectors in the Labelled Training Set which are the most discriminant (according to the selected criterion) among the vectors in the chosen dictionary.	1

References

Label	***Title***
[1]	N. Saito, R.Coifman, F.B. Geshwind, and F. Warner, Discrminant feature extraction using empirical probability density estimation and a local basis library, Pattern Recognition, 2002

11.vii Nearest Neighbor Classifier

Short Description

Basic nearest neighbor classifier.

Detailed Description

Nearest neighbor classifier assigns to a point the label that appears most often among the k nearest neighbors of the point. It is a very basic classifier, but it asymptotically (in the number of points) close to the Baysian classifier. It is quite efficient in low dimension, but in general quite enreliable in high dimensions, since it is very avversely affected by noisy distance computations.

Parameters

Label	***Description***
Estimation Method	Specifies how many nearest neighbors are used to vote for the label of a test point
Metric	Specifies which metric to use to find the nearest neighbors. Either the standard euclidean metric in the dimension of the data or the "maximum" metric (the distance between two points being the maximum of the distances of the projection on any coordinate axis) can be specified. • euclidean:use Euclidean metric. • maximum:use maximum or L^\infty metric.

Inputs

Name	***Description***	***Multiplicity***
Labelled Training Set	This is the set of labelled training points. To estimate the label of a test point, the nearest points in this training set are determined, and the label most represented among them is chosen as the label of the test point.	1
Classify	These are the datacubes to classify.	0

Outputs

Name	***Description***	***Multiplicity***
Classifications	These are the classification datacubes: the i-th slice of the datacube represents the probability that the point belongs to class i.	Classify

References

Label	*Title*
[1]	http://www.cs.sunysb.edu/~algorith/files/nearest-neighbor.shtml
[2]	http://www.physik3.gwdg.de/tstool/index.html

11.viii Nearest Neighbor Projection

Short Description

Maps to the set of distances from training classes.

Detailed Description

Given a training set consisting of k training classes, and a set of test vectors, this algorithms computes the map that maps each test vector in the vector of distances from each of the k training classes. Various options for which distance to use are given, including nearest point distance, further point distance, average distance, distance from approximate linear subspace and so on. The choice of distance is application, data set and objective dependent, and in general it greatly affects the result of the algorithm

Parameters

Label	*Description*
Point to set metric	Specifies the metric to use when measuring distances between a point x and a set A. • min:d(x,A) is the distance (measured according to <Metric>) between x and the closest point to x in A. • max:d(x,A) is the distance (measured according to <Metric>) between x and the farthest point to x in A. • mean:d(x,A) is the average distance (measured according to <Metric>) between x and the points in A. • center:d(x,A) is the distance (measured according to <Metric>) between x and the center of A. • subspace:d(x,A) is the norm of the component of x orthogonal to the subspace spanned by A. The subspace spanned by A is defined by A and some tolerance (e.g. because of noise), and is the one spanned by the top principal components of the datapoints in A truncated by tolerance..

Estimation method	Specifies how many nearest neighbors are used to vote for the label of a test point
Metric	Specifies which metric to use to find the nearest neighbors. Either the standard euclidean metric in the dimension of the data or the "maximum" metric (the distance between two points being the maximum of the distances of the projection on any coordinate axis) can be specified. • euclidean:Standard Euclidean metric. • maximum:Maximum or L^\infty metric: the distance between two points is given by the biggest of the absolute values of the differences of the coordinates of the two points..

Inputs

Name	*Description*	*Multiplicity*
Labelled Training Set	This is the set of labelled training points. To compute the NN-projection of a test point, the nearest points in this training set are determined, and the i-th coordinate of the result point is the distance from the closest training point in class i.	1
NN-project	These are the datacubes to be NN-project-ed.	0

Outputs

Name	*Description*	*Multiplicity*
NN-Projection Map	This is the nonlinear NN-Projeection Map learnt from the training set. It can be applied to any data (living in a corectly dimensioned ambient space).	1
NN-Projections	These are the NN-projections of the datacubes chosen in the input "NN-project".	NN-project

References

Label	*Title*

11.ix Spectra Normalization

Short Description

Normalizes the spectra of a data cube in various ways.

Detailed Description

Basic algorithms can be used to normalize the spectra in several ways.

Parameters

Label	*Description*
Normalization Type	Type of normalization to be applied to all the spectra in the datacube, one by one. • Linear01:Maps the range of each spectrum linearly onto [0,1].. • L2:Normalizes each spectrum so that its Euclidean length, or energy, is equal to 1.. • Mean + L2:Normalizes each spetrum so that its average is 0 and its Euclidean length, or energy, is equal to 1..

Inputs

Name	*Description*	*Multiplicity*
Datacubes	The data cubes whose spectra need to be normalized. The user can specify one or more data cubes.	0

Outputs

Name	*Description*	*Multiplicity*
Normalized Datacubes	The data cubes resulting from the normalization process.	Datacubes

References

Label	*Title*

11.x Principal Component Analysis

Short Description

Computes the principal components of the set of points

Detailed Description

Computes the principal components of the set of points [1,2,3]. These can be defined as the axes of maximum variance of the set of points. The algorithm is randomized in order to allow the computation for large datasets: a random subset is selected and its principal components are computed and returned. The size of the random subset is specified as a parameter by the user. The user also specifies the number of principal vectors to be computed. Because of randomization, if the random subset is small, in general one can expect a good relative estimate for the top eigenvectors, but not for the eigevectors except the few top on

Parameters

Label	***Description***
Number of PC's	Number of principal components to compute and return, expressed as a percentage of the total number of principal components (which is the same as the dimensionality of the space).
Size of random subset	Specifies the size of the random subset of points actually used for the computation, expressed as a percentage of the total number of samples available.

Inputs

Name	***Description***	***Multiplicity***
Datacubes	The datacubes whose principal components need to be computed, one datacube at a time.	0

Outputs

Name	***Description***	***Multiplicity***
SVD Basis	The principal components of the data set.	Datacubes

References

Label	***Title***
[1]	Golub, Matrix Computations
[2]	Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK User's Guide
[3]	Numerical recipes in C, www.nr.com

12 File Format Description

Hyperspectral Explorer can load files of the following types:

- Matlab files, properly formatted as specified below.
- HSE files: these are header files that are created when saving files from HyperSpectral Explorer in his own format, and is linked to Matlab files actually containing the data.
- PND files: these are files saved from NSTIS application by Plain Sight. Seamless integration between NSTIS actually allows to transfer automatically files from NSTIS application to HyperSpectral Explorer, via the File->Export command in the NSTIS application.
- ENVI files: these are a widespread format for hyperspectral images.

12.i Matlab files

Hyperspectral Explorer loads files saved from Matlab, with extension .mat, and satisfying the characteristics described below.

Let the filename be <Filename.mat>.
If a datacube is stored in the file <Filename.mat>, then this file should contain a Matlab variable called <Filename>. This variable should be a structure, containing a field named <Data>. This field contains the datacube values, as a 3 dimensional array, whose coordinates are X,Y,S, i.e. The two spatial coordinates first, and the spectral coordinate third.

If a training set is stored in the file <Filename.mat>, then this file should contain a Matlab variable called <Filename>. This variable should be a structure, containing a field named <Data> and a field named <Labels>. The field named <Data> should be a 2 dimensional array N by S, each row representing a spectrum in the training set. The field named <Labels> should be a 1 dimensional column vector of length N, the i-th entry representing the label (an integer number greater than 0) of the i-th spectrum in the field <Data>.

12.ii ENVI files

ENVI is a widespread file format for hyperspectral images. There are two ENVI files for each hyperspectral cube: a header file and a data file. The header file contains a description of the data, and other useful information (e.g. it may contain information about the frequency range, the date of collection etc...). The datafile contains the hyperspectral cube.
Hyperspectral Explorer will import ENVI files that have both header and the corresponding data file. The header should have extension .hdr, and the data file should have extension .img. When a file is loaded, a dialog with the header information of the

file is displayed.
The file is loaded into a Matlab structure that contains both the data and the description from the header file, which is thus accessible anytime by accessing the varaible from the Matlab command window Hyperspectral Explorer is connected to.

12.iii HSE files

These files are created when saving variables from the Hyperspectral Explorer workspace. These files are header files, and contain pointers (file names with path information) to Matlab files, as described above, that actually contain the data. If the Matlab files pointed to from inside the HSE file are moved or deleted or modified, the HSE file will become unusable.

12.iv PND files

These files are the default output when saving hyperspectral data from the NSTIS application by Plain Sight.

13 Error Description Table

ERROR	*CAUSE*	*SOLUTIONS*
Error E0001: could not connect to Matlab wrapper server.	The Matlab wrapper server, in the DLL file MatlabEngineWrapper.dll is not correctly registered.	Reinstall Hyperspectral Explorer. For the more advanced user: run regsvr32 <dir>/MatlabEngineWrapper.dll to re-register the Matlab wrapper server
Error E0002: could not connect to Matlab server (error code:-3)	The Matlab wrapper server failed to connect to the Matlab server. Either Matlab is not properly installed, or Matlab license server is not connected, or the user does not have enough privileges to connect to Matlab.	Reinstall Matlab and Hyperspectral Explorer. For the more advanced user: re-register the Matlab server manually. To do so, either read the instruction in the Matlab help, or simply run: Matlab /regserver from the Matlab binary directory to re-register the Matlab server.
Error E0003: failed to add the datagroup <name> to the datagroup server.	One of the data groups could not be created. The application definition file has been modified incorrectly.	Restore the original application configuration XML file AppDef.xml, or re-install HyperSpectral Explorer to restore it.
Error E0004: some resources could not be freed. You may have to shut down Matlab manually.	While disconnecting to Matlab and closing the application, HyperSpectral Explorer could not free all the objects allocated by Matlab, and/or could not disconnect from the Matlab engine.	If a “Matlab Command Window” is open, or present in the taskbar, close it manually. If Matlab does not close, go to the Task Manager (right-click on the taskbar, and select Task Manager), go to the “Processes” tab, and terminate the Matlab process.
Error E0005: failed to load algorithm (algorithm number xx).	At startup HyperSpectral Explorer connects to the various algorithms available. One such algorithm was not available, or its configuration file was not available or was not correctly formatted.	Restore the application definition file Appdef.xml and all the algorithm definition files in the Algorithms directory. If not available, reinstall HyperSpectral Explorer
E0006: failed to load the datagroups.	The application definition file Appdef.xml is not properly formatted.	Restore the original application configuration XML file AppDef.xml, or re-install HyperSpectral Explorer to restore it.